UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

Foodiefy: Descubre y comparte recetas

Apellidos: González Moreno
Nombre: Alberto
Titulación: Grado en Multimedia
Plan: 2009
Directora: Ana Gabriela Zúñiga Zárate
Fecha de lectura: 17 de julio de 2017

Índice

1. Resumen

Este proyecto consiste en la creación de un prototipo totalmente funcional de una aplicación móvil de recetas de cocina que sirva a modo de red social, tanto para conectar usuarios como para buscar y compartir recetas.

Con el fin de poder ofrecer un producto de calidad, para la realización de la aplicación han intervenido dos proyectos distintos; uno encargado de la parte de **diseño** y este otro encargado de la **programación**.

Se ha abordado el diseño de la arquitectura de los datos y de la propia aplicación, la autenticación segura por parte de los usuarios, la subida y búsqueda de recetas, el sistema de alertas, la página web de presentación, seguridad y usabilidad.

Se ha aplicado una metodología de trabajo ágil (*scrum*) para coordinar los dos proyectos mencionados. En el desarrollo de la aplicación, están presentes las tendencias y tecnologías más actuales que hay en el mercado con objetivo de seguir con la expansión y mejora de la aplicación a posteriori.

El resultado de este proyecto puede verse en www.foodiefyapp.com o buscando la aplicación en Google Play Store.

2. Palabras clave

App, Android, Java, Programación, Firebase, Cloud, Redes Sociales, Recetas, Usuarios

3. Glosario

A

alt

En HTML5 las imágenes pueden llevar un texto alternativo para mostrarlo si la imagen no está disponible o si se usa un lector de pantalla. 111

APK

"Android Package Kit" es el formato de archivos que usa Android para la distribución e instalación de apps. Contiene todo el código y recursos usado por la aplicación, "empaquetado" todo en un archivo. .65

app

Una aplicación móvil, applo o app (en inglés) es una aplicación informática diseñada para ser ejecutada en teléfonos inteligentes, tabletas y otros dispositivos móviles y que permite al usuario efectuar una tarea concreta de cualquier tipo. www.wikipedia.org 13

array asociativo

En programación, se le llama array asociativo a aquellos vectores o matrices en los que el índice no es númerico y ordenado, sino que el índice es variable y libre. Por ejemplo, el índice "myIndex" puede tener el valor "myValue". 55

B

backend

En informática, se le llama backend a aquella programación o servicios que son parte del servidor. Es decir, la interacción con bases de datos, administración, etc. 56

C

callback

Un callback es una "retrollamada", es decir 99

cuneiforme

Es una de las formas más antiguas de expresión escrita. Se realizaba mediante pictogramas representando palabras y objetos pero no conceptos abstractos. www.wikipedia.org 17

D

description

Ver *keywords*. 111

E

emojis

Es un término japonés para los ideogramas o caracteres usados en mensajes electrónicos y sitios web. www.wikipedia.org 42

H

handheld

El término handheld, hand-held computer o hand-helddevice, es un anglicismo que traducido al español significa.... 28

hardcodedado

Hard-code, término del mundo de la informática hace referencia a una mala práctica en el desarrollo de software que consiste en incrustar datos directamente (a fuego) en el código fuente del programa, en lugar de obtener esos datos de una fuente externa como un fichero de configuración o parámetros de la línea

4. Introducción

4.1. Formulación del problema

Hoy en día, la mayoría de gente joven tiene muy pocos conocimientos culinarios. A causa de esto, la mayoría mantiene unos hábitos alimentarios poco saludables.

> *[…] young people lack confidence in their preparation and cooking skills, not being 'trusted in the kitchen' to fend for themselves. Although parents are the main food preparers of these young people, they lack faith in their parent's ability to cook.*
>
> Extracto de los resultados del artículo *'I'm not trusted in the kitchen': food environments and food behaviours of young people attending school and college*.

A raíz de lo expuesto, en este proyecto se plantea una herramienta como solución a este problema. Se propone una *app* [1]donde los usuarios puedan tanto buscar recetas a partir de los ingredientes de los que disponen, como subir ellos mismos recetas a la *app*.

Además, como punto innovador, se busca que funcione como una red social, en el sentido de que los usuarios puedan comentar, puntuar y compartir aquellas recetas que han probado. De esta manera se crea una comunidad, la cual es la base de la *app* ya que el contenido es generado por los usuarios.

El *target* [2] de la app está dirigido a dos tipos de personas: aquellas que no tienen conocimientos de cocina y no saben qué cocinar (o no tienen el tiempo para pensar qué cocinar) y a aquellas personas que sí tienen conocimientos y desean compartirlos (subiendo recetas). Ambos grupos comparten interés por el mundo *fitness* y *healthy*.

Este grupo de personas suelen comer a menudo los mismos grupos de ingredientes, por lo que la opción de búsqueda de recetas basada en ingredientes es ideal para ellos. Como punto añadido, si se introducen valores nutricionales la aplicación gana valor y podría llegar a usarse a modo de seguimiento de dieta.

Un ejemplo de uso de esta *app*, sería:

> *Juan llega a casa después de su larga jornada de trabajo. Son las 21.00 de la noche y no ha pensado qué va a cenar. Juan tiene dos opciones: comer un plato*

[1] Una aplicación móvil, applo o app (en inglés) es una aplicación informática diseñada para ser ejecutada en teléfonos inteligentes, tabletas y otros dispositivos móviles y que permite al usuario efectuar una tarea concreta de cualquier tipo. www.wikipedia.org
[2] Público objetivo

precocinado, el cual no es saludable ni aporta los nutrientes necesarios, o bien puede cocinar él mismo algo con los ingredientes que dispone en casa. Juan decide cocinar él mismo, porque es una persona con hábitos saludables, pero no dispone de los conocimientos necesarios. Para ello, Juan sacaría su Smartphone [3]del bolsillo, abriría la app Foodiefy, introduciría manualmente los ingredientes que ha encontrado en su nevera y despensa, elegiría una receta que le propone la aplicación y podría cenar de manera saludable.

Juan ha preparado un plato que le ha propuesto Foodiefy y le ha añadido su toque personal. Juan podría dejar un comentario en la receta diciendo lo mucho que le ha gustado y lo bien que le queda un toque de limón. Adicionalmente, podría puntuar la receta para que así los otros usuarios pudieran ver el nivel de aceptación que tiene.

Al día siguiente, Juan llama a su madre y le pregunta por la receta de croquetas que hacía cuando él era un crío. Entonces decide compartir esa receta con el mundo, para ello abre la app Foodiefy y añadiría la receta de su madre.

[3] El teléfono inteligente (smartphone en inglés) es un tipo de teléfono móvil construido sobre una plataforma informática móvil, con mayor capacidad de almacenar datos y realizar actividades, semejante a la de una minicomputadora, y con una mayor conectividad que un teléfono móvil convencional. www.wikipedia.org

4.2. Definición de objetivos

El objetivo principal del proyecto es la creación de una *app*. Este objetivo se podría desglosar en las siguientes tareas:

- Crear una *app*
    - Definir un problema
        - Resolverlo
    - Definir una imagen corporativa
        - Crear un logotipo
        - Crear una línea de imagen
    - Analizar competencias y productos similares
    - Definir un plan de marketing
        - Desarrollar un plan de comunicación
            - De qué forma se atraen clientes
            - Cómo explicamos qué hace la *app* y para qué sirve
        - Definir acciones a corto/medio/largo plazo
            - Cómo dar a conocer la *app* y en qué medios realizar las distintas acciones (tv, redes sociales, radio, prensa)
            - Potenciar el uso de la *app*
    - Definir el diseño de la *app*
        - Diseño gráfico
        - Diseño de interacción
        - Diseño de *UI*[4]
        - Diseño de *UX*[5]
    - Creación de los grafismos necesarios para las acciones definidas en el plan de marketing
    - Definir una estructura interna de la *app*
        - Definir una base de datos
        - Definición de clases y métodos y sus relaciones
    - Conocer Firebase, integrarlo en un proyecto e intentar exprimir al máximo sus funcionalidades
        - Hosting, Database, Cloud Functions, Storage, Cloud Messaging

[4] La interfaz de usuario es el medio con que el usuario puede comunicarse con una máquina, equipo, computadora o dispositivo, y comprende todos los puntos de contacto entre el usuario y el equipo. www.wikipedia.org

[5] UX Design (User Experience Design) o “Diseño de Experiencia de Usuario” es una filosofía de diseño que tiene por objetivo la creación de productos que resuelvan necesidades concretas de sus usuarios finales, consiguiendo la mayor satisfacción y mejor experiencia de uso posible con el mínimo esfuerzo.

- Programar la app
  - Programación de los Modelos
  - Programación de los Adaptadores
  - Programación de las diferentes Activities
  - Programación de los Servicios necesarios
  - Aplicar diseño
- Programar la página web
  - Maquetación básica
- Publicar mi primera aplicación en Google Play Store

5. Estado del arte

5.1. Recetas de cocina

Las primeras evidencias históricas de recetas de cocina datan del 1600 a.C. Estas recetas procedían del sur de Babilonia y estaban escritas sobre tablillas de barro en escritura cuneiforme [6]e idioma acadio[7]. En el imperio griego ya había escritores culinarios aunque ninguna de sus recetas ha llegado a nuestros días. El escritor romano Marco Gavio Apicio hizo uno de los primeros libros de recetas conocidos en el mundo occidental, el *De re coquinaria*.

Ilustración 1 - Manuscrito de De re coquinaira, uno de los libros de cocina de recetas del imperio romano. www.wikipedia.org

En el medievo, el primer recetario fue escrito por un alemán en el siglo XIII. La cocina española tiene evidencias históricas del 1324 en el *Llibre de Sent Soví*. Este libro constaba de más de doscientas recetas de cocina catalana, incluyendo un índice y una introducción. Las recetas entonces, se describían como una serie de instrucciones que se debían realizar.

A finales del siglo XIX, en América del Norte, Isabela Beeton escribe *Book of Household Management*, que es uno de los primeros recetarios modernos.

En las últimas décadas del siglo XX aparece una estructura diferente de recetas de cocina que no es ingredientes-procedimiento. Empiezan a surgir programas televisivos de cocina, revistas e incluso blogs.

5.1.1. Características

Las recetas de cocina generalmente se agrupan según el ingrediente principal, el país de procedencia, el tipo de preparación, etc. Todas las recetas incluyen los siguientes elementos:

- Denominación y origen

[6] Es una de las formas más antiguas de expresión escrita. Se realizaba mediante pictogramas representando palabras y objetos pero no conceptos abstractos. www.wikipedia.org

[7] Lengua semítica actualmente extinta, hablada en la antigua Mesopotamia principalmente por asirios y babilonios durante el II milenio a. C. www.wikipedia.org

- Tiempo de preparación y en algunas ocasiones la dificultad de preparación
- Lista de ingredientes necesarios acompañados de las cantidades
- Pasos a seguir para la elaboración

Adicionalmente, aunque no es común, se suele añadir las herramientas necesarias. Estas herramientas se deducen del procedimiento a seguir.

En los libros de cocina modernos se suele incluir una fotografía de la elaboración acabada, emplatada y decorada. También es común encontrar ingredientes alternativos si es que hay alguno que es raro o difícil de encontrar y el cómputo total de calorías y/o otros valores nutricionales.

5.1.2. Actualidad

En una investigación de campo es posible determinar de qué manera se buscan recetas. No es de extrañar que los más jóvenes (hasta 40 años) recurren directamente a Internet.

En cambio, los más maduros primero consultan en algún libro o revista que tengan a mano. Es cuando ya han mirado en libros y no han encontrado nada que recurren a Internet.

Cuando se busca una receta o bien se realiza mediante una búsqueda rápida en algún motor de búsqueda (como Google) o bien se busca en una app.

Existen multitud de páginas web dedicadas a ofrecer recetas de cocina según su procedencia, para niños, comida vegetariana y un largo etcétera. Si bien es cierto que muchas de ellas ofrecen contenido de calidad, también es cierto que no dejan de ser bases de datos cerradas de recetas de cocina.

Cuando se habla de apps de cocina, el panorama es muy similar al de las aplicaciones web. Hay multitud de apps, unas más bonitas que otras, unas más sencillas que otras, pero todas acaban ofreciendo lo mismo: una base de datos cerrada de recetas.

Al usar estas aplicaciones, tanto en móvil como en web, la sensación es muy parecida a la de buscar una receta en un libro en el sentido de que la plataforma “no está viva”. En otras palabras; no se siente pertenecer a nada. De nuevo, tan solo se siente como si se consultase una base de datos predefinida.

A parte de libros, revistas, webs y apps, también existen otras plataformas en las que encontrar recetas de cocina: en televisión.

Hay dos tipos de programas televisivos de cocina: los puramente "recetarios" y los concursos.

Ilustración 2 - Logo de MasterChef

Estos programas tienen mucha aceptación en el público y son muy demandados (hay programas de ese estilo en muchísimas cadenas de TV). Ofrecen un formato dinámico, familiar, entretenido y variado. Un ejemplo de cada tipo son "Torres en la cocina" y "MasterChef", ambos de TVE.

5.2. Sistemas Operativos

En el presente existen varios sistemas operativos para dispositivos móviles, como pueden ser:

- Android
- iOS
- Windows Phone
- Blackberry OS
- Otros

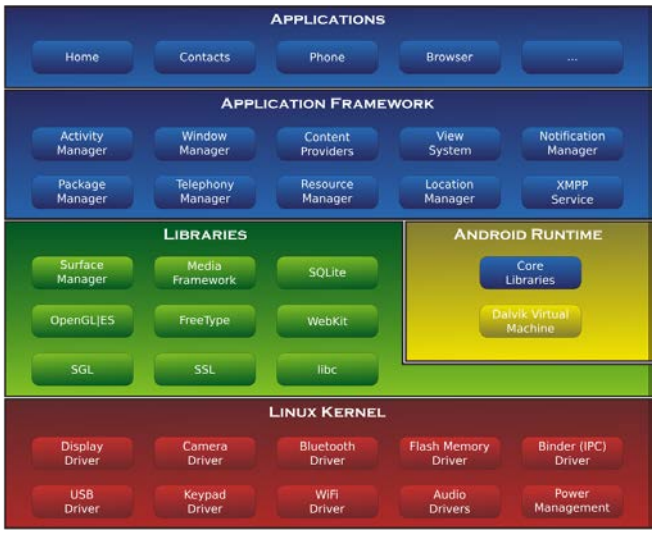


Ilustración 3 - Arquitectura del sistema Android, Wikipedia

Para la realización de este proyecto se ha elegido Android por ser *Open Source*[8] y disponer de una *SDK*[9] gratuita y accesible y ser el S.O. más popular actualmente.

Android es un S.O. basado en Linux, desarrollado por Android Inc., respaldada por Google en términos económicos para posteriormente, en el 2005, ser comprada por Google.

[8] El software de código abierto (en inglés open source software u OSS) es el software cuyo código fuente y otros derechos que normalmente son exclusivos para quienes poseen los derechos de autor, son publicados bajo una licencia de software compatible con la Open Source Definition o forman parte del dominio público. www.wikipedia.org
[9] Software Development Kit, kit de desarrollo de software

Versión	Nombre	API	Distribución
2.3.3-3.3.7	Gingerbread	10	1.0%
4.0.3-4.0.4	Ice Cream Sandwich	15	1.0%
4.1.x	Jellybean	16	4.0%
4.2.x		17	5.7%
4.3		18	1.6%
4.4	KitKat	19	21.9%
5.0	Lollipop	21	9.8%
5.1		22	23.1%
6.0	Marshmallow	23	30.7%
7.0	Nougat	24	0.9%
7.1		25	0.3%

Ilustración 4 - Versiones de Android, Febrero 2017

En el Informe Android de febrero de 2017 se aprecia que Lollipop y Marshmallow son actualmente las versiones con más presencia en la telefonía móvil con S.O. Android.

5.2.1. Evolución de Android

5.2.1.1. Donut

Con Android 1.6 aparece por primera vez el cuadro de búsqueda rápido, el cual ofrece resultados tanto de la web como de contenido local del dispositivo.

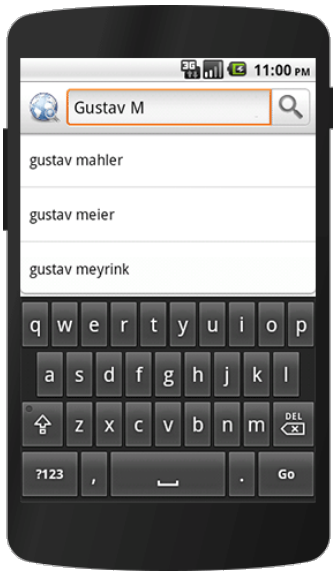


Ilustración 5 - Cuadro de búsqueda rápida. www.android.com

Además, gracias a las funcionalidades de Android Donut que permitía ejecutar el SO en diferentes resoluciones de pantalla, Android se encuentra disponible en diferentes formas y tamaños. Esto quiere decir que se permite a Android ser ejecutado en resoluciones distintas a la convencional (vertical, 320x480).

La store [10]de Android cambia de aspecto tanto por fuera como por dentro, y pasa de llamarse Android Market a Google Play Store.

5.2.1.2. Eclair

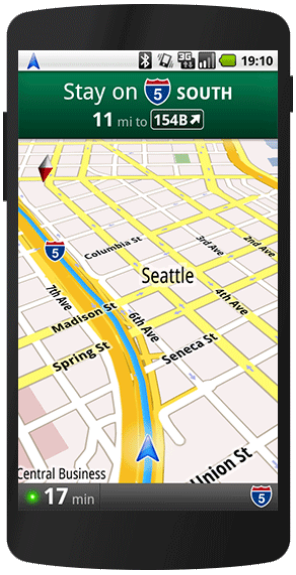


Ilustración 6 - Google Maps Navigation. www.android.com

En este momento los smartphones se redefinieron al ofrecer al usuario información GPS con datos de Google Maps. Los smartphones ofrecían lo mismo que un navegador GPS, pero de manera gratuita.

Android 2.1 no solo trajo consigo Google Maps Navigation, sino que introdujo más personalización al sistema operativo. Esto se consiguió con la incorporación de fondos de pantalla animados.

La síntesis de voz se introdujo en Android 2.1, permitiendo redactar un texto hablándole al dispositivo.

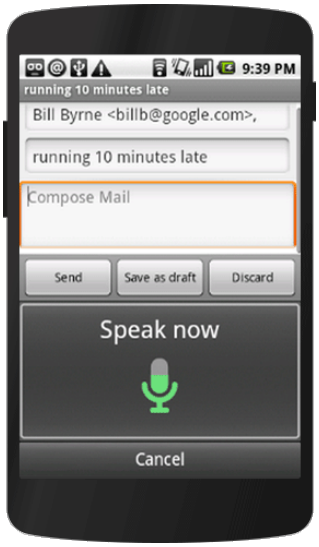


Ilustración 7 - Síntesis de voz. www.android.com

[10] Sitio en el que se distribuye software para dispositivos móviles. Del inglés, "tienda".

5.2.1.3. Froyo

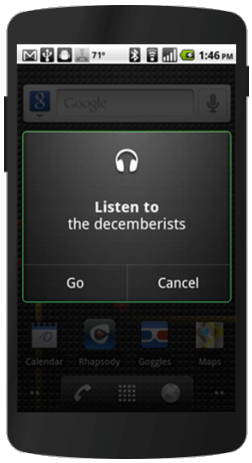


Ilustración 8 - Acciones de voz. www.android.com

Las funcionalidades de voz de Android se vieron mejoradas en Android 2.2 con las Acciones de voz que permitían realizar las acciones más básicas del teléfono. Estas incluían realizar búsquedas, obtener indicaciones, añadir alarmas, etc.

La conectividad se vio mejorada al introducir la Zona Wi-Fi portátil. Esta funcionalidad permitía convertir tu dispositivo móvil en una especie de router con el que conectarte a internet. De esta manera podías conectar un ordenador portátil a internet mediante tu Smartphone.

El rendimiento de Android se vio muy mejorado con la introducción del compilador Dalvik JIT (rendimiento x5 en el código) y también se introdujo el motor JavaScript V8 (x2-x3 rendimiento respecto a JavaScript).

5.2.1.4. Gingerbread

El sector del videojuego en dispositivos móviles dio un gran paso con la introducción de Android 2.3. Permitiendo el acceso al audio, controles del dispositivo, gráficos y almacenamiento, se ofreció la oportunidad a los desarrolladores de crear juegos 3D.

Apareció por primera vez la tecnología NFC (comunicación de campo cercano) con la que nació una nueva gama de aplicaciones que ofrecían un servicio distinto. Con esta tecnología se podía transmitir información entre dispositivos con tan solo acercarlos entre sí.

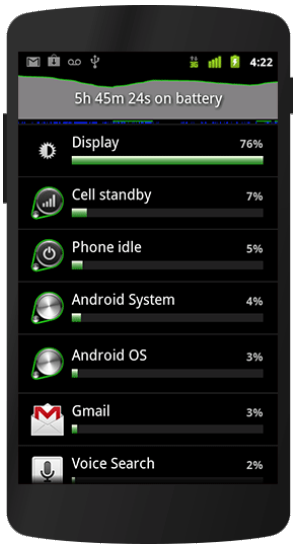


Ilustración 9 - Gestión de la batería. www.android.com

La gestión de la duración de la batería fue más llevadera en Android 2.3 ya que ofrecía información acerca del uso y consumo de la batería.

5.2.1.5. Honeycomb

Ilustración 10 - Ajustes rápidos. www.android.com

Con la introducción de las tablets al mercado, nace la necesidad de adaptar el diseño. Es por ello que en Android 3.0 se permite leer libros, ver vídeos o explorar mapas de una manera más fácil e intuitiva.

Atrás se quedan los botones físicos de navegación con la incorporación de una nueva barra en pantalla que incluía estos botones, de manera digital.

Honeycomb introduce los Ajustes rápidos, una manera rápida de acceder a información básica como la hora, la batería… o cambiar algún ajuste como el brillo de la pantalla o activar y desactivar el WiFi.

5.2.1.6. Ice Cream Sandwich

De nuevo, la capacidad de personalización crece con la introducción de carpetas y bandeja de favoritos. También se permite a los widgets, que insertan contenido de aplicaciones en la pantalla de inicio, cambiar su tamaño por lo que aumentó la flexibilidad.

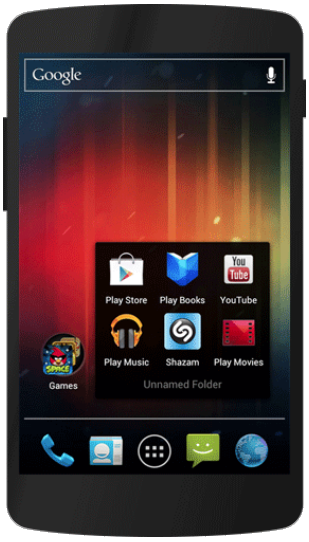


Ilustración 11 - Carpetas de aplicaciones. www.android.com

El control de uso de datos de redes móviles fue introducido en Android 4.0. Se permitía monitorizar el uso de datos, limitarlos el consumo e incluso inhabilitar la red.

Ilustración 12 - Uso de datos de redes móviles. www.android.com

Aparece Android Beam que usa la tecnología NFC introducida en Honeycomb para compartir información (contenido multimedia, contactos, aplicaciones…) con otro dispositivo Android sin la necesidad de menús o vincular los dispositivos.

5.2.1.7. Jelly Bean

Personalización. De nuevo es la premisa de Android. Por ello se introduce Google Now, un servicio que ofrece información totalmente personalizada a modo de asistente personal. Entre la información que ofrece se encuentra, entre otras, la información meteorológica o trayecto y duración para llegar a casa.

Algo novedoso que se introduce en Android 4.1 son las notificaciones accionables. Estas notificaciones permiten ser ampliadas para mostrar más información e incluyen botones de acción de la propia aplicación (como responder a un mensaje, o dar me gusta). De esta manera se abre una nueva forma más rápida de interactuar con el contenido de una notificación.

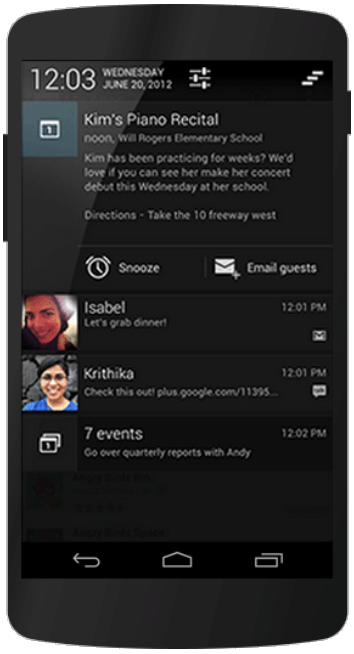


Ilustración 13 - Notificaciones accionables. www.android.com

Las tablets que corrían JellyBean permitían la funconalidad de Multiusuario (más tarde se incorporaría en smartphones con Android Lollipop). Esta funcionalidad es análoga a las cuentas de usuario de Windows, por ejemplo. De esta manera cada usuario tiene su espacio personalizado.

5.2.1.8. KitKat

Las funciones de voz de Android se ven mejoradas y se introduce la conocida función “Ok, Google”. Esta funcionalidad ofrece al usuario dictar una acción sin siquiera interactuar con el dispositivo, tan solo diciendo “Ok, Google” se activa el reconocimiento de voz.

KitKat introduce el diseño envolvente el cual permite al usuario sumergirse en la actividad que realiza ocultando todo lo demás.

Los teléfonos de vuelven un poco más inteligentes con esta versión 4.4 de Android. KitKat destaca los contactos con las que más interactúas y permite buscar sitios cercanos directamente en el teléfono.

5.2.1.9. Lollipop

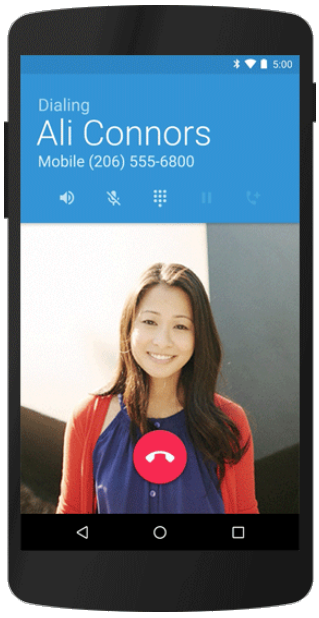


Ilustración 15 - Material Design. www.android.com

Google introduce el Material Design y Android se ve completamente renovado. La navegación se ve mejorada con esta nueva línea de diseño.

Lollipop introduce Multipantalla, con lo que se puede cambiar entre dispositivos y así reanudar en tu Android TV la canción que escuchabas en tu smartphone.

Ilustración 14 - Multipantalla. www.android.com

Las notificaciones aparecen en la pantalla de bloqueo en formato de tarjeta y los controles específicos permiten la personalización del contenido mostrado.

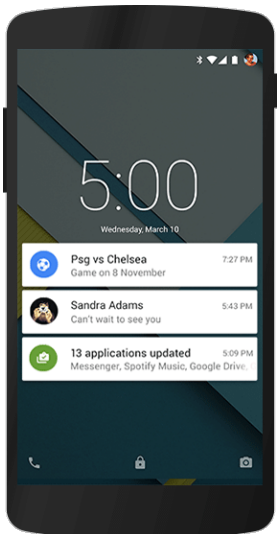


Ilustración 16 - Notificaciones en pantalla de bloqueo. www.android.com

5.2.1.10. Marshmallow

Google Now sigue presente, pero mejorada su accesibilidad en esta versión 6.0 de Android. Ahora es posible acceder a Google Now desde cualquier lugar, sin tener que abandonar la aplicación en uso. Tan solo es necesario mantener pulsado el botón de inicio.

La privacidad se ve mejorada al permitir qué permisos obtienen las aplicaciones. Es decir, qué información se comparte con las aplicaciones y la posibilidad de revocar esos permisos.

Marshmallow optimiza el dispositivo permitiendo una mayor duración de la batería ofreciendo las funciones Descanso y Aplicaciones inactivas.

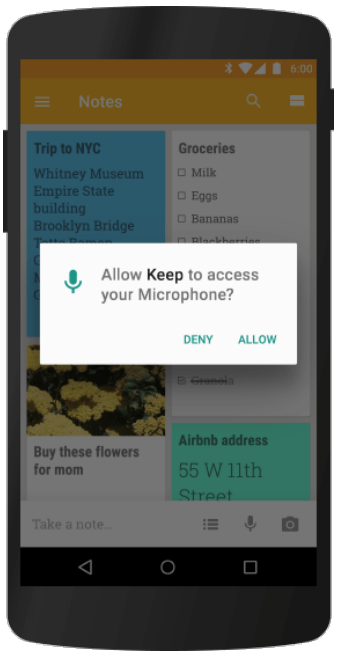


Ilustración 17 - Permisos de Android. www.android.com

5.2.1.11. Nougat

En Android 7.0 se introduce de manera nativa la función Multiventana. De esta manera existe la posibilidad de tener visible en pantalla más de una aplicación y poder moverse entre ellas de manera rápida y fácil.

También se mejoran las notificaciones, facilitando la agrupación de diferentes notificaciones de la misma aplicación.

El desplegable de configuración rápida admite más personalización y contiene más ajustes.

Como novedad, en Android N se introducen las Instant Apps. Estas aplicaciones son versiones reducidas y simplificadas de una aplicación para ser ejecutadas sin la necesidad de descarga ni instalación. De esta manera se puede compartir un enlace con un contacto y obtener información de una aplicación sin tenerla instalada.

5.3. Evolución de las aplicaciones móviles

Junio 1983 – Steve Jobs de Apple predice un nuevo sistema de distribución de software. En una conferencia en Aspen llamada “the future isn’t what it used to be”, Jobs propone un sistema de distribución similar a una tienda de música en los que se puede comprar a través de las líneas telefónicas.

Ilustración 18 - Psion Epoc

Enero 1987 – Un ordenador *handheld*[11], el Psion Epoc, usa un sistema operativo Symbian que contiene aplicaciones básicas como una agenda.

Agosto 1993 – Apple desarrolla una PDA (la Newton MessagePad) que contiene aplicaciones preinstaladas como un explorador web, email, calendario, libreta de direcciones, y reconoce texto escrito a mano.

Noviembre 1993 – Un artículo de “Business Week” predice que los futuros aparatos “de información” realizarán conexiones instantáneas a un mundo de entretenimiento, comunicación y dataos digitalizados.

[11] El término handheld, hand-held computer o hand-held device, es un anglicismo que traducido al español significa “de mano” (computadora o dispositivo de mano) y describe al tipo de computadora portátil que se puede llevar en una mano mientras se utiliza

Enero 1996 – La *Palm OS* es descrita como la primera PDA [12](aunque no fue la primera) porque sí fue la primera en "hacerlo bien". Lanzó una industria, caló en la cultura popular y preparó el terreno para una nueva era de dispositivos.

Ilustración 19 - Nokia 6110 con el juego Snake

Diciembre 1997 – El Nokia 6110 contenía el mítico juego Snake permitiendo a la gente llevar ese juego en el bolsillo.

Octubre 1999 – Se publica el *Wireless Application Protocol* (WAP), un estándar técnico para acceder a redes a través de redes inalámbricas en móviles.

Octubre 2001 – Sale al mercado la primera generación de iPods con aplicaciones preinstaladas como el *Solitaire and Brick* y ofreciendo "1000 canciones en tu bolsillo".

Abril 2003 – Apple lanza la *store* de música de iTunes (iTunes Music Store) con 200.000 canciones a 0.99USD cada una. Logró vender un millón de canciones en su primera semana.

Diciembre 2004 – Aparece el primer software malicioso para dispositivos móviles, el *SymbOS.Cabir*. Se distribuía a través de conexiones Bluetooth y distribuía archivos corruptos.

Febrero 2006 – Aparece la segunda principal instancia de software malicioso que enviaba mensajes SMS, el *Trojan.Redbrowser*.

Junio 2007 – Apple anuncia que los desarrolladores podrán crear aplicaciones Web 2.0 con aspecto y comportamiento similar a las aplicaciones preinstaladas de iPhone. Estas webapps podrían acceder a los servicios de iPhone, incluyendo localización, llamadas… En el mismo año, se lanza el primer iPhone en EEUU vendiendo 270.000 unidades durante las primeras 30 horas.

Marzo 2008 – Apple anuncia en un evento cómo las aplicaciones de terceros podrían ser creadas de manera nativa para iPhone.

[12] Asistente personal digital

Ilustración 20 - Logo App Store

Julio 2008 – Se lanza la App Store (iOS) con 552 apps, de las cuales 135 eran gratuitas. En una semana se realizan 10 millones de descargas y más de 800 aplicaciones nativas están disponibles para descargar.

Setiembre 2008 – En 60 días desde el lanzamiento de la App Store, se realizan más de 100 millones de descargas y hay disponibles más de 3.000 aplicaciones en 62 países. Sale al mercado *Fitbit*, el primer *wearable*[13].

Octubre 2008 – Se lanza el Android Market de Google y se convierte en el segundo principal distribuidor de aplicaciones móviles y el rival de la App Store de Apple. A la vez, Google lanza el *Googlephone*, el HTC Dream, el primer Smartphone que usa Android.

Enero 2009 – Apple lanza una campaña comercial conocida como "there's an app for that" (en español, hay una app para eso). Pronto se convierte en un slogan.

Abril 2009 – Se lanza el Blackberry World Store y se convierte en el tercer principal distribuidor de aplicaciones móviles, rivalizando con Apple y Google.

Noviembre 2009 – Se lanza WhatsApp.

Diciembre 2009 – Se lanza Angry Birds, la aplicación de pago número 1 en iTunes en 68 países.

Enero 2010 – ZunZuneo Cuban es lanzada al mercado. Es una red social creada por el gobierno de EEUU que permitía a los cubanos comunicarse entre ellos con las restricciones de Internet impuestas por el gobierno.

Ilustración 21 - Logo de la store de Windows

Octubre 2010 – Windows lanza la Windows Phone Store y se convierte en el cuarto principal distribuidor de aplicaciones móviles.

[13] Dispositivo electrónico que se lleva sobre, debajo o incluido en la ropa. Son un buen ejemplo del Internet of Things

Diciembre 2010 – Aparece el software malicioso más sofisticado hasta la fecha para Android. El *Geinimi* tenía el potencial de tomar el control absoluto del móvil, una vez infectado, a través de un servidor remoto.

Marzo 2011 – Se lanza Amazon App Store.

Junio 2011 – Zynga Games pierde 34 millones de usuarios. Sus base de usuarios son los jugadores de FarmVille, los cuales empiezan a migrar del juego de Facebook a otros juegos de sus smartphones.

Setiembre 2011 – La eliminación de un juego satírico de la App Store (iOS) reabre el debato sobre cómo Apple trata de manera diferente a las apps respecto a la música, libros o películas. El juego incluía referencias a la explotación infantil, suicidios de trabajadores de fábricas…

Octubre 2011 – Una firma de seguridad digital, S21sec, descubre *ZeuS MitmO*, un importante *malware*[14].

Diciembre 2011 – Apple anuncia que la revolución de las apps ha creado más de 290.000 empleos en EEUU desde la introducción de iPhone en 2007.

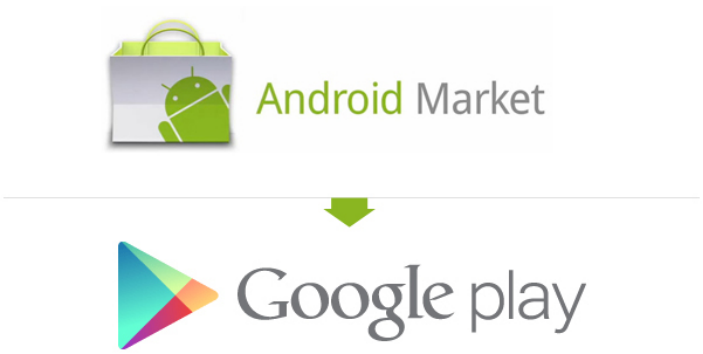


Ilustración 22 - Cambio en la store de Android

Marzo 2012 – La store de Android, el *Android Market*, cambia de nombre a Google Play Store como parte de la última actualización de Android 2.2.

Abril 2012 – Facebook adquiere Instagram por la cantidad de 1.000 millones de USD, la cantidad más alta pagada por una compañía de apps.

Mayo 2012 – Angry Birds llega a los 1.000 millones de descargas.

Noviembre 2012 – Se lanza Candy Crush para iOS.

Diciembre 2012 – El término “app” llega a su pico en búsquedas de Google.

[14] Es un tipo de *software* que tiene como objetivo infiltrarse o dañar una computadora o sistema de información sin el consentimiento de su propietario

Mayo 2013 – La App Store (iOS) alcanza los 50.000 millones de descargas. Apple publica la lista de las apps más descargadas. Angry Birds es la app de pago más descargada, mientras que Facebook lidera las apps gratuitas.

Julio 2013 – Google Play alcanza los 50.000 millones de descargas. Hay disponibles más de un millón de apps.

Octubre 2013 – La App Store alcanza el millón de apps.

Noviembre 2013 – Snapchat rechaza la oferta de 3.000 millones de USD ofrecidos por Facebook.

Enero 2014 – eMarketer predice que habrá 1.750 millones de smartphones a causa del avance de las redes 3G y 4G y el abaratamiento de los mismos. Según estudio de Nielsen, los usuarios estadounidenses mayores de 18 años emplean un 65% más de tiempo en apps que dos años antes.

Febrero 2014 – Facebook adquiere WhatsApp por la cantidad de 19.000 millones USD.

Ilustración 23 - Logo Facebook

Marzo 2014 – El 97% del malware en dispositivos móviles se encuentra en Android, pero sólo el 0.1% de las aplicaciones de Play Store (Android) está infectada. La mayoría viene de stores ajenas, predominantemente en el Medio Este y Asia.

Mayo 2014 – Snapchat mueve al día unas 700.000 millones de fotos. Snapchat sigue creciendo pese al rechazo de la oferta de Facebook. Android lanza el Android Wear y se convierte en el principal dispositivo wearable en el que los desarrolladores empiezan a hacer apps.

Junio 2014 – La App Store está presente en 155 países, se calcula que genera más de 15.000 millones USD. Google presenta una normativa de diseño enfocado a Android, el Material Design.

Agosto 2014 – Se produce lo que se ha descrito como “un ataque muy dirigido” en el que se filtran más de 500 fotografías iCloud de gente famosa y se distribuyen en el sitio web 4chan.

Setiembre 2014 – El 20.3% de las descargas de la App Store son videojuegos, un 10.36% son apps didácticas y el 1.9% son redes sociales.

Octubre 2014 – WhatsApp es usado como servicio público de información por la BBC distribuyendo mensajes de audio y alertas de texto e imágenes para ayudar a parar combatir el Ébola. Se descubre en el mismo mes que una app llamada Flash Torch, con más de 50 millones de descargas, roba información sensible del usuario tales como detalles bancarios.

5.4. Las aplicaciones móviles

En la actualidad se puede concluir, basándose en datos de Yeeply:

- 22% apps descargadas solo se usan una vez
- 60% apps descargadas se eliminan al cabo de un mes
- 80% del tiempo de uso de tablet/móvil es con una app dentro de las categorías: entretenimiento, productividad, local, compra
- Solo el 1% de las apps son rentables (económicamente hablando)

¿Qué tienen de especial ciertas aplicaciones?

Candy Crush

Ha revolucionado la manera en que se juega integrándose con la plataforma Facebook haciendo que más gente juegue y la que juega más aún.

Ilustración 24 - Logo CandyCrush

Evernote

No ofrece nada especial respecto a su competencia. ¿Su diferencia? Es multiplataforma.

Ilustración 25 - Logo Evernote

Tinder

Esta app ha revolucionado la manera de conocer gente nueva. ¿Cómo? Tiene tres puntos fuertes: Casi nulidad de configuración a causa de la integración con Facebook, geolocalización y simplicidad de uso.

Ilustración 26 - Logo Tinder

De esta manera, ¿cuáles son las principales características que debe tener una app?

- Integración con plataformas o herramientas
- Posibilidad de utilización en más de una plataforma (por ejemplo: móvil y PC)
- Geolocalización
- Simplicidad

Remitiéndose al informe anual que publica App Annie, en 2016 se realizaron 90.000 millones de descargas entre las dos principales *stores*. Esta cifra supone un aumento del 15% respecto al año anterior, 2015.

Ilustración 27 - Crecimiento de las descargas de aplicaciones móviles. www.appannie.com

En 2016, entre App Store y Google Play, obtuvieron en ingresos 35.000 millones de USD, lo que supone un aumento del 40% respecto al 2015.

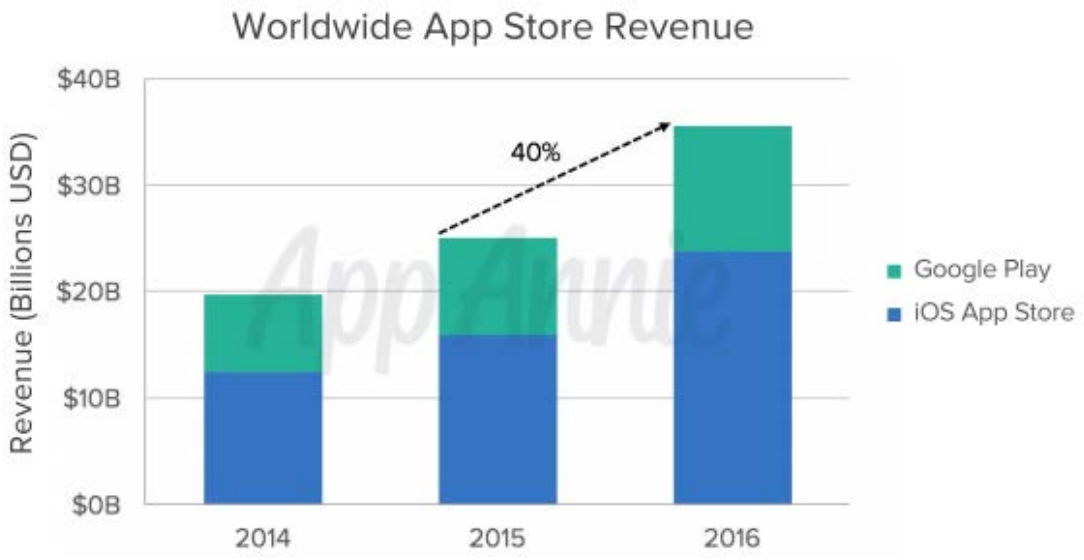


Ilustración 28 - Tasa de ingresos en las principales stores. www.appannie.com

En España, durante el 2016, estas fueron las apps más descargadas (combinando Android e iOS):

Top Apps of 2016: Spain Combined
iOS and Google Play Downloads

Rank	App	Company
1	WhatsApp Messenger	Facebook
2	Facebook	Facebook
3	Facebook Messenger	Facebook
4	Wallapop	Wallapop
5	Instagram	Facebook
6	Snapchat	Snap
7	Spotify	Spotify
8	Skype	Microsoft
9	Clean Master	Cheetah Mobile
10	AliExpress	Alibaba Group

Ilustración 29 - Top apps en descargas, 2016. www.appannie.com

Y en el mismo período, según sus ingresos:

Top Apps of 2016: Spain Combined iOS and Google Play Revenue

Rank	App	Company
1	Spotify	Spotify
2	Netflix	Netflix
3	Tinder	InterActiveCorp (IAC)
4	LOVOO	LOVOO
5	Badoo	Badoo
6	Meetic	InterActiveCorp (IAC)
7	AdoptAGuy	GEB AdoptAGuy
8	Sing! Karaoke	Smule
9	Skype	Microsoft
10	Sweat with Kayla	Kayla Itsines

Ilustración 30 - Top apps en ingresos, 2016. www.appannie.com

Así pues, según estos datos facilitados por App Annie, se concluye que las apps más descargadas en España son WhatsApp, Facebook y Facebook Messenger (las tres propiedades de Facebook) mientras que las apps qué más ingresos obtuvieron son Spotify, Netflix y Tinder (las dos primeras de contenido bajo demanda y la tercera una app social).

5.5. Tecnologías en aplicaciones móviles

Actualmente se pueden crear aplicaciones móviles nativas (iOS, Android) y otras emuladas, como puede ser *Phonegap* de *Apache Cordova* o *Titanium* de *Appcelerator*, que permiten simular el funcionamiento de una aplicación local.

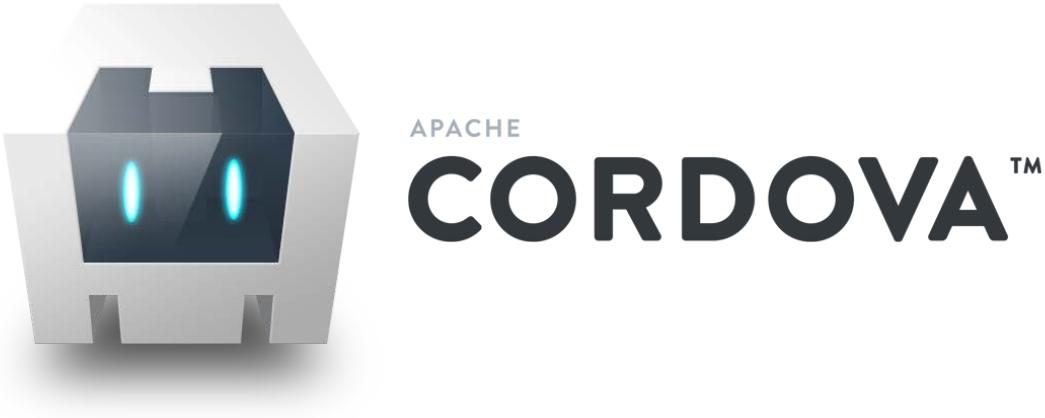


Ilustración 31 - Logo Apache Cordova

Para programar una aplicación iOS se puede hacer en los lenguajes *Swift*, y *Objective C*. Mientras que en Android se pude hacer en *C*, *C++* y *Java*. Para hacer una aplicación emulada se usa *HTML* combinado con *JavaScript*, y básicamente el navegador web accede a los *gadgets* del dispositivo, como el acelerómetro.

Ilustración 32 - Logo de Objective C

5.5.1. App nativas

Este tipo de aplicaciones son aquellas que han sido programadas en lenguajes específicos del sistema operativo de destino. Es decir, Objective C, Swift para iOS y C, C++ y Java para Android. Otros sistemas operativos usarán diferentes lenguajes, por ejemplo .NET para Windows Phone.

Hacer apps nativas tiene sus ventajas y contras.

Ventajas:

- Se obtiene el mayor rendimiento en el dispositivo
- Se obtiene un look&feel más óptimo acorde al sistema operativo

    - A causa de esto se obtiene una mejor experiencia de usuario
- Se obtiene acceso a todas las capacidades del dispositivo
- Se siente que “funciona bien” al usarla y se siente más como una app y no una web
- Facilidad de incorporación en las stores

Flaquezas:

- Requiere un lenguaje de programación específico para cada plataforma
- Suele resultar más complejo y difícil su desarrollo comparado con una aplicación híbrida.

5.5.2. Apps híbridas

Es la manera más sencilla de hacer una aplicación mediante HTML y CSS combinado con JS. Si bien es la más fácil, también es la que mayor portabilidad ofrece dado que es independiente de la plataforma. No todo lo que reluce es oro. Las aplicaciones híbridas son las que peor rendimiento ofrecen.

Fortalezas:

- Multiplataforma
- Sencillez en el desarrollo

Debilidades:

- Rendimiento
- La UI no es propia del sistema operativo

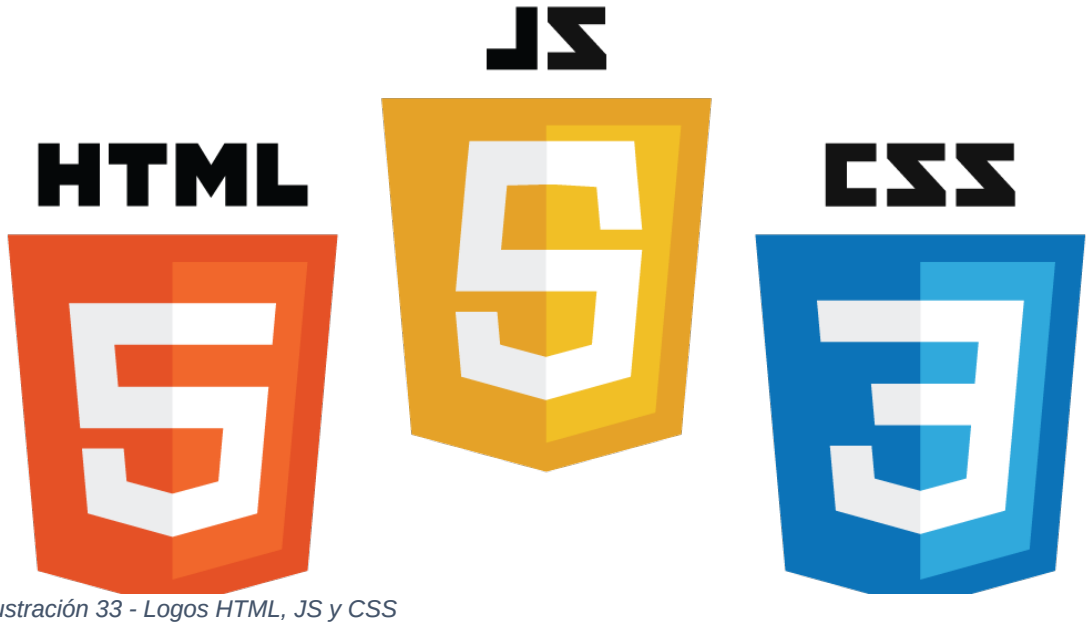


Ilustración 33 - Logos HTML, JS y CSS

5.5.3. Apps generadas

En los últimos años han surgido ciertas herramientas que “generan” apps nativas, como Xamarin. Esto quiere decir que se escribe una app en el lenguaje propio de Xamarin y luego se generan las distintas versiones para cada plataforma a partir del código ya escrito. “Write once, Run everywhere, AND Be Native?” es el eslogan de Xamarin.

Puntos fuertes:

- Comparte código entre distintas plataformas
- La curva de aprendizaje es menor en comparación a las apps nativas
- Cada vez más desarrolladores se suman a este flujo de trabajo
- Obtiene un rendimiento casi igual al de una app nativa

Sus flaquezas:

- Pese a compartir código, es necesario programar en lenguaje específico para cada sistema operativo para acceder a ciertos *gadgets* o servicios.
- La estructura del proyecto es compleja
- Requiere disponer de un Mac para compilar para iOS y un PC para compilar para Windows Phone.
- Son tecnologías jóvenes que requieren madurar

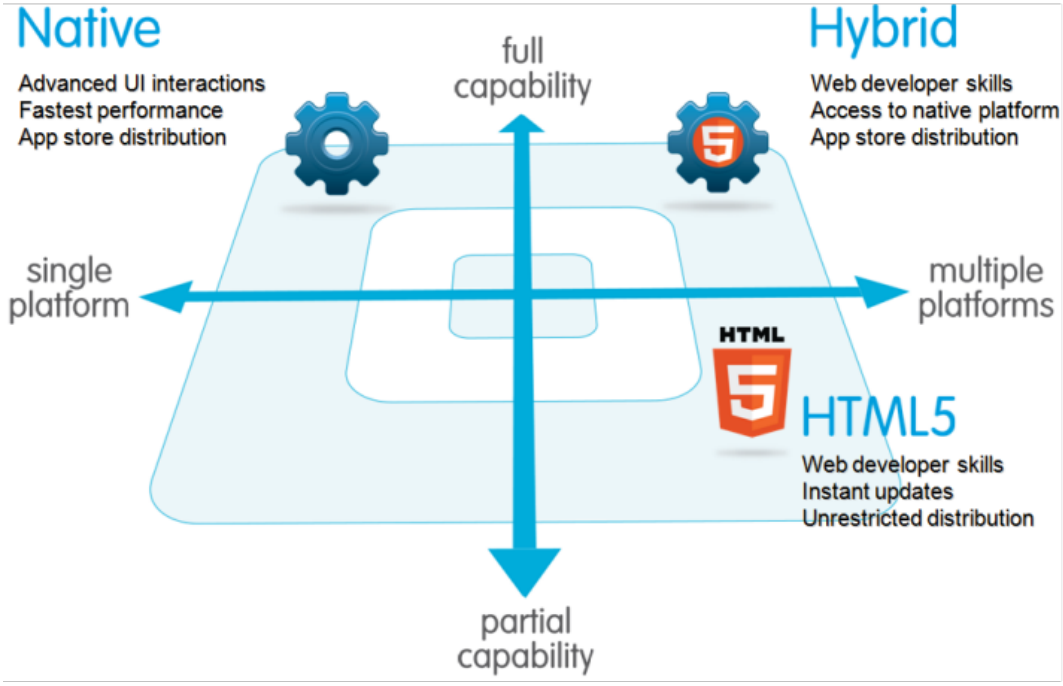


Ilustración 34 - Apps nativas vs híbridas. https://developer.salesforce.com

5.6. Almacenamiento de datos en aplicaciones móviles Android

Las aplicaciones, normalmente, harán uso de algún sistema para obtener información y así poder mostrar datos. Existen varias formas de almacenar datos:

- En bases de datos
- En archivos del dispositivo, tanto en la memoria interna como la externa
- En preferencias del dispositivo

Android es compatible de forma nativa con bases de datos *SQLite*. Todas las clases de la aplicación pueden acceder a la base de datos, pero clases de otras aplicaciones no tienen acceso.

Si se guarda información en preferencias compartidas, esta información es almacenada en pares de clave-valor, persistente entre sesiones incluso si se detiene la aplicación. Son accesibles desde distintas aplicaciones.

Se permite generar y guardar archivos en la memoria interna del dispositivo o una tarjeta de memoria. Esta información es privada y no es accesible desde otras aplicaciones (ni el usuario tampoco tiene acceso).

5.7. Hábitos de consumo

Referenciando al informe de *Ditrendia*, publicado en 2016, *Mobile en España y en el Mundo 2016*, se extraen las siguientes conclusiones:

Ilustración 35 - Logo Ditrendia

- En España, los *smartphones* representan el 87% de todos los teléfonos móviles.
- La venta de *tablets* cae en picado en todo el mundo (10% menos que el año anterior, 206.8 millones). No obstante, las ventas crecen en España. Actualmente en 3 de cada 4 hogares españoles hay, al menos, una *tablet*.
- En el mundo, más de la mitad de las visitas que reciben los principales buscadores, se realizan a través de un móvil. El 62% del tiempo online procede de *smartphones* y *tablets*.
- La mayoría de descargas de *apps* proceden de las propias *stores*, seguidas de recomendaciones de amigos (62%) y de redes sociales (30%).

- Los pagos son cada vez más comunes. Para los españoles, el 42% ha pagado al menos una vez por una *app*.
- El perfil medio que hace uso de *apps* es de joven de entre 25 y 34 años, de ciudad.
- En España, Android es el sistema operativo más utilizado y común, representando un 84% del total en 2015. Seguido de iOS, que supone un 12%, y Windows, con tan solo un 3%. Samsung es la marca más vendida, seguida de otras marcas chinas.
- El uso de *apps* en España supone el 89% del tiempo total dedicado a *smartphones*.
- Los usuarios de entre 18 y 34 años dedican, de media, 29.6 horas cada mes a las redes sociales.
- El 91% de usuarios accede a redes sociales desde el móvil.
- La mayoría accede a través del móvil a partir de las 16.00 de la tarde, y desde una *tablet* a partir de las 20.30 de la noche.
- La mensajería instantánea ha modificado cómo se realizan las campañas de marketing. El uso de *emojis*[15] ha aumentado un 777%.

[15] Es un término japonés para los ideogramas o caracteres usados en mensajes electrónicos y sitios web. www.wikipedia.org

5.8. SQL vs NoSQL

SQL [16]es un lenguaje específico de gestión de bases de datos relacionales (*Structured Query Language*). Por el contrario, *NoSQL* [17]quiere decir “no sólo SQL” (*Not Only SQL).*

Los sistemas *NoSQL* son sistemas de gestión de bases de datos **no relacionales**. Hay diversas maneras para almacenar información:

- En documentos
- En pares clave-valor
- Familias de columnas
- Gráficos

Para este proyecto, acorde a las necesidades y características expuestas más adelante, se hará uso de un sistema *NoSQL* basado en documentos.

Primero se debe entender cómo funciona un *Document-Oriented Database;* los datos se almacenarán en documentos, los documentos similares se guardan en colecciones.

De forma análoga:

SQL	NoSQL
Fila	Documento
Tabla	Colección

Tabla 1 - Analogía entre SQL y NoSQL

En una base de datos relacional, se gestiona con lenguaje SQL. En cambio, en NoSQL, se gestiona con *queries* [18]de *JavaScript* dado que los documentos almacenados tienen un formato *JSON*[19].

[16] Lenguage de consultas, del inglés “Structured Query Language”. Da acceso un sistema de gestión de bases de datos relacionales que permite especificar diversos tipos de operaciones en ellos. www.wikipedia.org
[17] En informática, NoSQL (a veces llamado "no sólo SQL") es una amplia clase de sistemas de gestión de bases de datos que difieren del modelo clásico de SGBDR (Sistema de Gestión de Bases de Datos Relacionales, en inglés, RDBMS) en aspectos importantes, siendo el más destacado que no usan SQL como lenguaje principal de consultas. www.wikipedia.org
[18] En informática, se llama query a la consulta realizada a una base de datos. Es la interacción entre la aplicación y la base de dato.
[19] JavaScript Object Notation. Manera en que se definen 'objetos' en JavaScript.

Ilustración 36 - Ejemplo de sintaxis en SQL y NoSQL, www.sitepoint.com

A continuación, se exponen las principales diferencias entre los dos sistemas de gestión de bases de datos.

SQL	NoSQL Document-Oriented Database
Uso adaptado y extendido, mayor soporte. (A causa de llevar más tiempo en el mercado)	Administración poco usable, generalmente mediante comandos por consola.
Uso de tablas	Documentos con estructura *JSON*
Uso de cláusula ***JOIN***	No existen los *JOINS*
Poco escalable	Gran **escalabilidad** horizontal
Asegura la **integridad** de los datos; si se espera recibir un campo de texto y se está recibiendo un número, la propia base de datos lanzará un error.	No existe el mecanismo para asegurar la integridad de los datos, esto provoca un mayor **rendimiento**.
Información estructurada, jerarquías. Facilita el posterior análisis de datos.	Grandes cantidades de información, adaptación a las necesidades.
Más rápido en agregación.	Escritura de datos más rápida.
Data templates (Dificultan la posibilidad de error). Relacionado con la integridad de los datos.	**Flexibilidad**, pero puede llevar a inconsistencias.
Data Schemas (Se define de antemano la estructura de la base de datos, antes de empezar a producir. Índices, Claves primarias, Claves foráneas…)	*Schemaless* (Los datos se pueden añadir en cualquier sitio, en cualquier momento.)

Normalization. Minimiza la reduncancia de datos mediante referencias entre tablas.	*Denormalization*. Aunque se pueden referenciar documentos, puede optarse por repetir información. Esto repercute en el tamaño de los archivos, pero también ofrecen unas *queries* más rápidas pero más lentitud al actualizar varios registros a la vez.
Reglas para la integridad de los datos. La base de datos protege la integridad, de manera que lanzará un error si se quiere borrar un campo que está referenciado en otra tabla.	No existe el mecanismo.
Transacciones (o todo, o nada).	No existen transacciones como tales. Las modificaciones de los documentos se realizan de manera atómica; es decir, si actualizamos tres valores, o se actualizan los tres satisfactoriamente o no se modifica el documento. Aunque no hay equivalente a las transacciones SQL, existen métodos que se programan para llevar a cabo ese comportamiento.
Lenguaje declarativo estandarizado	*Queries* de *JavaScript* en formato *JSON*.

Tabla 2 - Tabla comparatida entre SQL y NoSQL

5.9. Firebase

Firebase es una plataforma de desarrollo móvil y web app. Fue comprada por Google en 2014 y desde entonces ha ido mejorándose hasta lo que es hoy. Esta plataforma ofrece soluciones para aplicaciones web, Android e iOS. Ofrece una API relativamente sencilla, integrada en un solo SDK.

Entre las funcionalidades que integra Firebase se encuentra, entre otras:

- Base de datos en tiempo real
- Autenticación
- Notificaciones
- Soluciones de almacenamiento
- *Analytics*
- Monetización

Sólo con ver lo que ofrece esta plataforma, y que sea de Google, ya tiene muchos puntos positivos para ser una buena candidata de ser integrada en cualquier aplicación moderna.
No obstante, es la solución de base de datos en tiempo real lo que la hace realmente atractiva para *Foodiefy*.
Firebase Realtime Database funciona de igual manera que lo haría una base de datos *NoSQL* de tipo *Document Oriented Database*. Es decir, almacena un gran *árbol JSON* con diferentes *nodos*. Firebase es realmente **muy rápido**.
Además, con la funcionalidad de *Authentication* se oferta una manera *sencilla* de implementar el sistema de usuarios dentro de *Foodiefy*.

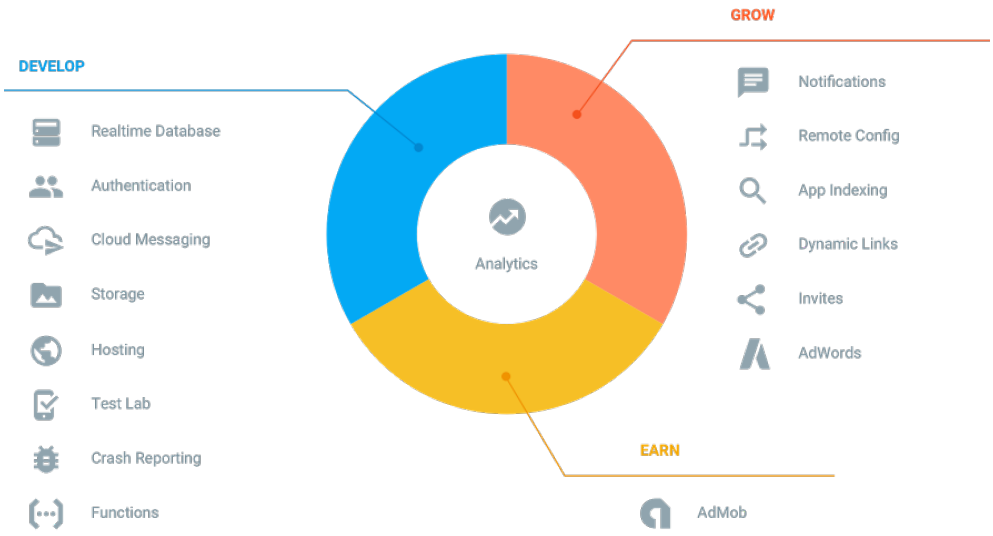


Ilustración 37 - Servicios de Firebase

5.10. Git

Git es un sistema de control de versiones [20]desarrollado por Linus Torvalds (creador del *kernel* [21]de Linux). Git ofrece la posibilidad de llevar un control de todas las modificaciones que se realizan en los archivos.

Además de controlar, también ofrece la posibilidad de revertir los cambios hacia un punto exacto del pasado. Git funciona por línea de comandos aunque existen *softwares* para realizar estas operaciones con una interfaz gráfica. Para este proyecto se hace uso de *Sourcetree*.

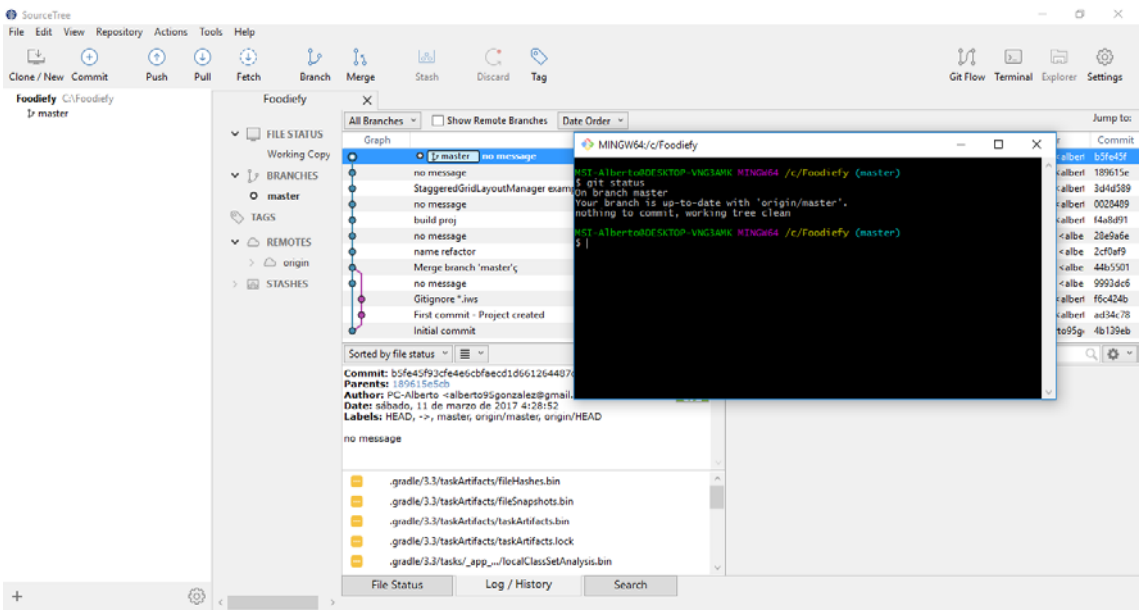


Ilustración 38 - Captura de pantalla del programa Sourcetree con una ventana de línea de comandos de Git.

En la captura anterior, de una fase temprana de desarrollo, se observan los diferentes *commits* realizados sobre la rama *master*. En la interfaz se observan las acciones básicas de Git (Commit, Push, Pull, Merge).

[20] Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Una versión, revisión o edición de un producto, es el estado en el que se encuentra el mismo en un momento dado de su desarrollo o modificación. www.wikipedia.org
[21] Núcleo de un Sistema Operativo

5.11. Android Studio

Para desarrollar una aplicación Android, existen diferentes opciones como son, entre otras:

- AppInventor
  - Visual, sin código. Sencillo, simple y gratuito.
- Netbeans
  - Sin soporte nativo para Android, se pueden desarrollar aplicaciones Java.
- Android Studio
  - Soporte nativo para Android, soporte para desarrollar *XML Layouts* [22]mediante un editor visual, con *drag&drop*, autocompletación y sugerencias de código…

Ilustración 39 - Captura de pantalla del emulador de Android integrado en Android Studio.

Por lo expuesto, y porque es Android Studio el programa usado en clase de *BETMA VII: Programació en Android* impartida en el octavo semestre del Grado en Multimedia, se ha elegido desarrollar *Foodiefy* en Android Studio.

Android Studio integra emuladores para ver, de una manera rápida y sencilla, la aplicación que se está desarrollando.

[22] En informática y diseño, se conoce como *layout* a la disposición de los elementos dentro de la interfaz gráfica. Podría definirse como una "plantilla" en la que se disponen elementos. En Android, los *layouts* se definen en formato *XML*.

5.12. Análisis de la competencia

Cuando un usuario busca una receta, lo primero que hace es buscar en Google. Desde los resultados de búsqueda navega entre diferentes páginas que ofrecen recetas. Si no la busca en Google ni la busca en un libro de recetas, usa una aplicación móvil.

Actualmente en las Stores existen aplicaciones de recetas de cocina. Algunas son muy parecidas a nuestra idea.

Recetario Villy:

Ilustración 40 - Logo Recetario Villy

Esta app ofrece recetas subidas por los usuarios y permite realizar búsquedas basadas en categorías o en ingredientes. La opción de búsqueda por ingrediente no se parece a la idea que propone Foodiefy. En el caso de Villy, es un campo de texto donde se introduce una cadena de caracteres y se realiza una búsqueda de esa cadena. La precisión no es comparable a la que se obtendría en Foodiefy ya que no se realizaría la búsqueda con un campo de texto, sino con una especie de sistema de *tags*, uno por ingrediente, y se buscarían recetas que contengan exactamente esos *tags*.

Hatcook:

Ilustración 41 - Logo Hatcook

Esta aplicación es mucho más parecida a Foodiefy que la anterior. Permite subir y buscar recetas basadas en ingredientes. Las recetas se pueden comentar, puntuar y compartir. Tal vez la flaqueza de esta aplicación es que su diseño es un poco liado y que no tiene un target definido. Además **no es una aplicación nativa**. Esta aplicación sería el competidor directo de Foodiefy.

Nestlé Cocina. Recetas y Menús

Nestlé dispone de una aplicación web y otra aplicación móvil.

Ilustración 42 - Logo Nestlé Cocina

Esta aplicación ofrece un sistema de búsqueda de recetas basado en ingredientes, igual que Foodiefy y Hatcook. La interfaz que presenta la aplicación móvil se percibe sobrecargada y llena de elementos innecesarios que pretenden añadir valor a la app pero en cambio se lo quitan. Tiene muchas opciones y se percibe un poco caótica. Las recetas son de una base de datos estática, como las anteriores apps.

SuperCook

Ilustración 43 - Logo SuperCook

SuperCook es una aplicación web que ofrece ni más ni menos que un motor de búsqueda de recetas basado en ingredientes que dispones. Eso es todo lo que ofrece, aunque es exactamente el motor de búsqueda que ofrece Foodiefy. SuperCook es de habla inglesa, está muy bien posicionado (aparece en primer lugar en Google al buscar por “search recipes”). Ofrece una interfaz sencilla e intuitiva.

5.12.1. Conclusión

Se concluye que pese a la existencia de productos similares con fuerza y peso en el mercado, no se cubren todas las necesidades que Foodiefy cubre. Ninguna de ellas está pensada como red social, ni como una plataforma para compartir recetas. Todo al contrario, sólo permiten la búsqueda de recetas (a excepción del Recetario Villy).

Ninguna de las aplicaciones analizadas cumple todas las expectativas de Foodiefy, solo algunas de ellas.

Es por ello que Foodiefy puede tener grandes oportunidades al ofrecer una comunidad donde compartir, un buen sistema de búsqueda de recetas e información nutricional específica de la receta. Todo ello, junto a un diseño simple, minimalista y efectivo, convierten a Foodiefy en una aplicación mejor que las que hay actualmente en el mercado.

6. Tecnologías empleadas

Para la realización del proyecto se emplearán las siguientes tecnologías y herramientas:

- Android Studio
- Firebase
- Git con UI mediante el software SourceTree

La elección no es arbitraria, sino que viene dada por las siguientes razones.

Se usará Android Studio por el soporte nativo con Android y el emulador que incorpora. Se desarrolla en Android porque es un sistema operativo libre, hay mayor posibilidad de entrar en el mercado (el cual es mayor), hay menores problemas de privacidad...

Firebase ofrece todas las herramientas que Foodiefy necesita: base de datos NoSQL, almacenamiento, usuarios, autenticación, notificaciones y analytics. Es por ello que se ha elegido Firebase.

Para facilitar la accesibilidad al código fuente se hace uso del sistema de control de versiones Git.

6.1. Espacio de trabajo

Para el desarrollo del proyecto, se ha trabajado en dos ordenadores personales diferentes. Uno es un portátil y otro un sobremesa.

Las características principales del portátil son:

- Procesador Intel Core i7 6700HQ
- 16Gbytes de memoria RAM DDR4-2133
- Tarjeta gráfica nVidia GTX 940MX
- Windows 10 Pro x64

Las características principales del sobremesa son:

- Procesador Intel Core i5 3570K
- 16Gbytes de memoria RAM DDR3-1333
- Tarjeta gráfica nVidia GTX 760 x2 (SLI)
- Windows 10 Pro x64

En ambos equipos se ha trabajado con las mismas versiones de software:

- Java 1.7, durante el desarrollo se actualizó a la versión 1.8
- Android Studio 2.3.2, con versión de Gradle 2.3
- Node.js 6.9.5

- Firebase 3.7
- Git 2.11.1.windows.1
- SourceTree 2.0.20

6.2. Firebase Database

Firebase Database es una base de datos NoSQL en tiempo real. En este proyecto ha tenido una gran relevancia, pues desde el principio se buscaba una base de datos eficiente, escalable.

Cuando se trabaja con Firebase Database, se añaden unos *Listeners* a ciertas rutas en la base de datos. Estos *listeners* "escuchan" cambios en las rutas definidas. Hay dos tipos de listeners: para "escuchar" una sola vez, o constantemente. Para habilitar la persistencia de datos y que esta sea más robusta, se ha usado en el proyecto el segundo tipo de *listener*.

Cuando los datos en la base de datos cambian, se reciben varios eventos. Uno de ellos es el "onDataChange". A continuación se muestra un sencillo ejemplo:

```
DatabaseReference database = FirebaseDatabase.getInstance().getReference();

database.child("ingredients").addValueEventListener(new ValueEventListener()
{
	@Override
	public void onDataChange(DataSnapshot dataSnapshot) {
		//Codigo, código, código
	}
	@Override
	public void onCancelled(DatabaseError databaseError) {
		//Código, código, código
	}
});
```

Primero se define una referencia a la base de datos. Una vez definida, se añade un *listener* en la ruta */ingredients/*.

En el código de arriba se definen dos eventos, uno para cualquier tipo de cambio en la base de datos y otro para tratar cualquier tipo de error.

La base de datos devuelve *snapshots*, que son como fotografías del momento exacto de la base de datos. En estas *snapshots* se encuentran los datos. En este caso, la *snapshot* está definida con nombre "dataSnapshot" y contendría toda la información que

hay en la ruta /ingredients/. En esta ruta están todos los ingredientes de la base de datos, conteniendo ID, nombre y recetas asociadas.

Para realizar la subida de datos se han usado básicamente dos métodos: setValue() y updateChildren().

El método *setValue* sobrescribe todo el nodo de la base de datos con la información que se le dé. Es decir, si se quisiera añadir a nuestro directorio de ingredientes una receta asociada a ese ingredient, la ruta sería /ingredients/ingredient_id/recipes/. Si se aplicase el método setValue sobre esa ruta, se sobrescribirían las recetas asociadas con la nueva receta. Por eso no es válido este método en este caso concreto, y se usa *updateChildren.* A continuación se muestra un ejemplo aplicado al proyecto:

```
DatabaseReference rootRef = FirebaseDatabase.getInstance().getReference();

rootRef.updateChildren(updateObject).addOnCompleteListener(new
OnCompleteListener<Void>(){

	@Override

	public void onComplete(@NonNull Task<Void> task) {

		if (task.isSuccesful()) {

			//Se ha actualizado correctamente

		}

		else {

			//Algo ha fallado

		}

	}

});
```

En el código de arriba se define una referencia a la base de datos. A diferencia del ejemplo anterior, en este caso no se navega a ningún directorio, sino que se realiza la acción de actualizar sobre la raíz. Este ejemplo es el código que se usa en la aplicación para subir una receta. Sobre la referencia al directorio raíz, *rootRef*, se aplica el método *updateChildren* pasándole como parámetro el valor con el que se quiere actualizar, en este caso *updateObject*.

updateObject es un *array asociativo* [23] en el que a cada índice es la ruta en la base de datos correspondiente. A continuación se expone el código correspondiente:

```
Map<String, Object> updateObject = new HashMap<>();
updateObject.put(“recipes/” + id, recipe);
updateObject.put(“users/” + mUser.getUid() + “/recipes/” + id, true);
for (Map.Entry<String, String> child : recipe.ingredients.entrySet()) {
      updateObject.put(“ingredients/” + child.getKey().toString() +
      “/recipes/” + id, true);
}
```

Lo primero que se hace es declarar el objeto *updateObject* y a continuación se le añade en el índice “recipes/id_receta” el valor *recipe* que es otro objeto del tipo Recipe que contiene todos los datos. Seguidamente se le añade en el índice “users/id_usuario/recipes/id_receta” el valor true. Por último, se recorre en un bucle del tipo *for* todos los ingredientes de la receta y se añaden al índice “ingredients/id_ingrediente/recipes/id_receta” el valor true.

De esta manera se ha generado un solo objeto que contiene toda la información para actualizar diferentes nodos de la base de datos en una misma operación.

El método *setValue*, que sobrescribe todo lo que haya en el nodo, se ha usado, por ejemplo, en el siguiente extracto del código:

[23] En programación, se le llama array asociativo a aquellos vectores o matrices en los que el índice no es númerico y ordenado, sino que el índice es variable y libre. Por ejemplo, el índice "myIndex" puede tener el valor "myValue".

```
DatabaseReference alertRef = FirebaseDatabase.getInstannce().getReference();

alertRef.child("users").child(FirebaseAuth.getInstance().getCurrentUser().get
Uid()).child("alerts").child(id).setValue(null).addOnCompleteListener(new
OnCompleteListenner<Void>(){

      @Override

      Public void onComplete(@NonNull Task<Void> task) {

            if (task.isSuccessful()) {

                  //Correcto

            }

      }

});
```

Este código se ejecuta cuando deslizamos una alerta hacia el lado. A parte de eliminarla de la interfaz del usuario, también se elimina de la base de datos. Primero se crea una referencia a la base de datos (*alertRef*) y se navega hacia el directorio */users/user_id/alerts/alert_id/* y se le asigna el valor *null* (*setValue(null)*). FirebaseDatabase interpreta el valor *null* como inexistente, por lo tanto elimina el nodo.

6.3. Firebase Authentication

Firebase Authentication proporciona servicios de *backend* [24]y una API sencilla de usar. Para acceder a la base de datos (tanto lectura como escritura) o a los servicios de Storage, es necesario que el usuario haya iniciado sesión y esté autenticado.

Por defecto, cuando se entra en Foodiefy se autentica e inicia sesión de manera anónima, de esta forma le damos al usuario la posibilidad de navegar por la aplicación y acceder a la base de datos sin necesidad de iniciar sesión.

Para iniciar sesión, primero recibe los credenciales de autenticación, en nuestro caso un *token* OAuth [25]de un proveedor federado, concretamente Google. Más tarde, estos credenciales se pasan al Firebase Authentication que verifican los credenciales y devuelven una respuesta al cliente. Si la respuesta es positiva, el usuario queda autenticado y es posible acceder a los datos de perfil básicos del usuario, tales como su nombre o dirección de correo.

[24] En informática, se le llama backend a aquella programación o servicios que son parte del servidor. Es decir, la interacción con bases de datos, administración, etc.

[25] Open Authorization es un estándar abierto. Presenta un protocolo de autorización segura de una API de modo estándar y simple para aplicaciones web o móvil. www.wikipedia.org

En el apartado 7 (Seguridad) se encuentra información más detallada sobre el protocolo OAuth.

A continuación se expone el código usado para comprobar la autenticación del usuario:

```
FirebaseAuth mAuth = FirebaseAuth.getInstance();

FirebaseUser mUser = mAuth.getCurrentUser();
```

Como se puede comprobar, es muy simple. Primero se recoge una instancia de Firebase Authentication y después el usuario.

Hay varias posibilidades:

- mUser es *null*, en este caso el usuario no se ha autenticado
- mUser no es *null*, pero no se ha autenticado con Google. En este caso mUser.isAnonymous() es igual a *true*.
- mUser no es *null* y además mUser.isAnonymous() es *false*. En este caso el usuario se ha autenticado con unos credenciales válidos.

Se pueden añadir *Listeners* a la instancia de FirebaseAuth (igual que en el caso de Firebase Database) para detectar cambios en el estado de la instancia. Es decir, para detectar si el usuario inicia o sale de sesión como se ve a continuación:

```
mAuth.addAuthStateListener(new FirebaseAuth.AuthStateListener() {

      @Override

      Public void onAuthStateChanged (@NonNull FirebaseAuth firebaseAuth) {

            mUser = mAuth.getCurrentUser();

      }

});
```

6.4. Firebase Storage

Con Firebase Storage se le da la opción al usuario de guardar contenido generado por él mismo en la nube. En Foodiefy, se usa para almacenar las imágenes de las recetas.

Está perfectamente integrado con Firebase Authentication, por lo que no es posible acceder a los datos sin estar debidamente autenticado.

Se puede tanto subir archivos a la nube como descargarlos para su visualización.

Firebase Storage funciona de manera parecida a Firebase Database, en el sentido de que los archivos se almacenan en directorios. Por ejemplo, en este proyecto hay un

directorio */images/* en Firebase Storage, que dentro tiene una carpeta para cada receta. De esta manera, es sencillo acceder a la imagen asociada a una receta con la siguiente ruta:

/images/id_receta/imagen

A continuación se ve el código respectivo a la parte de subir la imagen asociada a la receta:

```
StorageReference sRef;

sRef = FirebaseStorage.getInstance().getReference();

sRef = sRef.child(“images/”).child(id).child(fileName);

UploadTask uploadTask;

uploadTask = sRef.putBytes(imageByteArray);

uploadTask.addOnSuccessListener(new
OnSuccessListener<UploadTask.TaskSnapshot> (){

      @Override

      Public void onSuccess (UploadTask.TaskSnapshot taskSnapshot) {

            //Código, código, código

      }

});
```

Básicamente, lo que se hace es definir una referencia al directorio y en ella se define un *UploadTask* que es “la tarea a realizar”. Este UploadTask es aplicar sobre la referencia el método *putBytes* pasándole como parámetro un array de bytes (*imageByteArray*). Se le añade un *listener* (igual que en Firebase Database y Authentication) que recibe, igual que en Firebase Database, una *snapshot* pero en este caso no es de la base de datos sino de la propia “tarea a realizar”, UploadTask. Este *snapshot* contiene toda la información necesaria, como la URL de la imagen.

6.5. Firebase Cloud Messaging

Con Firebase Cloud Messaging existe la posibilidad de enviar mensajes y notificaciones a un dispositivo cliente de la app.

Este servicio se usa desde la interfaz gráfica de Firebase, por lo que no sirve para este proyecto ya que se quieren enviar notificaciones automáticas. Aun así es necesario integrarlo en el proyecto para poder recibir notificaciones. Estas notificaciones se enviarán desde Firebase Cloud Functions.

A continuación se muestra el código correspondiente al Servicio para recibir los mensajes y notificaciones.

Este servicio debe extender de *FirebaseMessagingService* y debe sobrescribir el método *onMessageReceived*.

```
@Override

public void onMessageReceived (RemoteMessage remoteMessage) {

                //Código

}
```

El método *onMessageReceived* recibe un objeto *RemoteMessage* el cual contiene toda la información. Se puede acceder a ella a través de los métodos *getNotification()* y *getData()*.

6.6. Firebase Cloud Functions

Firebase Cloud Functions permite ejecutar código en el servidor en respuesta a eventos. Estos eventos pueden estar provocados por peticiones HTTP o por características de Firebase.

A diferencia de toda la programación que se ha realizado hasta ahora (Java), estas funciones que se ejecutan en el *backend* de Firebase están en JavaScript. Firebase Cloud Functions soporta el nuevo estándar ECMAScript 6 (ES6), por lo que se puede escribir JavaScript usando las nuevas funcionalidades que ofrece ES6.

Un ejemplo de estas funciones, en Foodiefy, es la siguiente:

Se ha definido una función *updateScores*. Cuando se define una función en Firebase Coud Functions que responde a un evento y no a una petición HTTP, como es el caso, se hace sobre un directorio de Firebase concreto. En otras palabras; *updateScores* es una función definida sobre la base de datos Firebase Database en el directorio */recipes/{recipe_id}/score/{score_id}* que responde a un evento.

Esta función responde al evento *ref.write*, es decir, cuando se escribe sobre el directorio. Cada vez que un usuario puntúa una receta, esta puntuación se añade en el directorio y salta el evento. La función hace la media de todas las puntuaciones y añade un nodo *scoreTotal* en el nodo padre de la referencia, es decir; */recipes/{recipe_id}/scoreTotal/.*

Además de añadir la puntuación media, también envía una notificación al usuario. Una vez enviada la notificación, se añade una “copia” de esa notificación en la base de datos para poder mostrar las alertas al usuario ya que no es posible obtener un “registro” o “histórico” de notificaciones enviadas/recibidas.

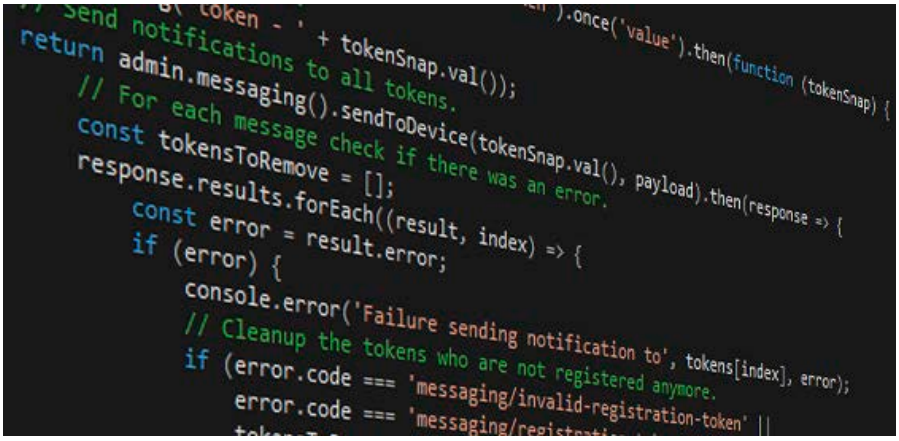


Ilustración 44 - Fragmento de código en el que se envía una notificación tras recibir un comentario en una receta. JavaScript.

6.7. Firebase Hosting

Firebase Hosting es capaz de ofrecer una web app de manera fácil, sencilla y gratuita. Funciona a través de la línea de comandos de Firebase y se realiza una conexión segura SSL.

En Firebase Hosting estará alojada la *landing page*[26] para Foodiefy inicialmente, mientras que más adelante se convertirá en *web app* ofreciendo así otra plataforma para usar Foodiefy ya que al pertenecer al propio proyecto, tendrá acceso a la misma base de datos y usuarios.

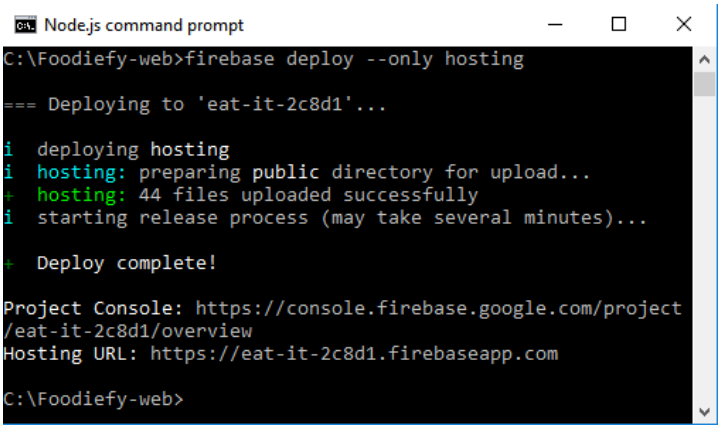


Ilustración 45 - Ventana de línea de comandos desplegando una página web a los servicios de Firebase Hosting

https://www.foodiefyapp.com

[26] Se le llama página de aterrizaje o landing page a aquellos websites a los cuales se llega a través de un enlace, banner o anuncio. Generalmente acostumbran a extender la información del anuncio, promocionar algo, etc.

7. Seguridad

Firebase usa el protocolo SSL con claves de 2048 bits para los certificados. Este protocolo proporciona conexiones de red seguras encriptadas.

A parte de usar un protocolo seguro, la base de datos está protegida por las Firebase Realtime Database Rules. Estas reglas determinan si el usuario que quiere acceder a los datos puede hacerlo o no, cómo pueden estructurarse los datos o cómo indexar los datos. Todas las solicitudes de lectura y escritura pasan primero por las reglas de seguridad y solo se llevan a cabo si estas lo permiten.

La medida de seguridad principal es no permitir el acceso ni de escritura ni de lectura a la base de datos si el usuario no está autenticado (puede estar autenticado de manera anónima).

Las reglas de seguridad se definen en formato JavaScript, para no permitir el acceso a usuarios no autenticados se definen de la siguiente manera:

```
{
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null",
    "ingredients": {
      "$iid": {
             ".indexOn": ["name", "recipes"]
      }
    },
    "users" : {
      "$uid" : {
        ".indexOn" : "recipes"
      }
    }
  }
}
```

Adicionalmente, se ha especificado un índice en el nombre de los ingredientes y las recetas asociadas y las recetas de un usuario.

7.1. Análisis de seguridad

Se ha realizado una pequeña auditoría de seguridad en la que se ha analizado el tráfico de paquetes, acceder al código fuente o a datos sensibles.

Primero de todo se han intentado simular peticiones HTTP sin necesidad de la app. Algunos motivos por los que se hacen son realizar acciones que la app no ofrece por sí misma; cambiar puntuaciones, editar descripciones, acceder a datos no visibles desde la app, etc.

En una primera prueba se ha usado la siguiente configuración:

- Móvil Android con *root*
- App Xposed: permite instalar módulos para modificar el sistema siendo *root*
- Módulo JustTrustMe para Xposed: permite descifrar el tráfico HTTPS cuando se usa un sistema de seguridad SSL Pinning.

La técnica de SSL Pinning es una técnica de seguridad para evitar ataques *man-in-the-middle*, o en otras palabras; evitar que un tercero intercepte información por el camino. Se trata de revocar certificados de seguridad válidos si no son de nuestro servidor. Normalmente se aplica en situaciones que requieran un alto nivel de seguridad como son PayPal, Google, aplicaciones bancarias, etc. También es habitual encontrar esta técnica en aplicaciones como Pokemon Go para que los usuarios no puedan *hackear* el juego, o al menos que sea muy difícil.

A pesar de poder descifrar alguna petición, no se ha podido obtener ninguna petición que realiza la app.

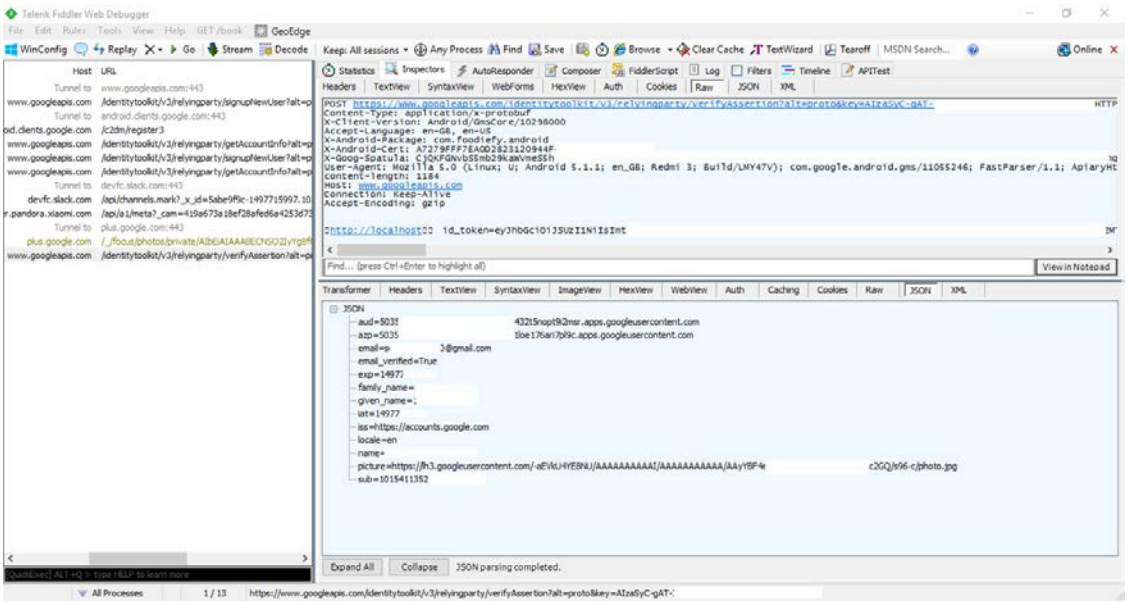


Ilustración 46 - Descifrado de peticiones con el software Fiddler

Para una segunda prueba se ha añadido a la configuración anterior el módulo Inspeckage para Xposed que permite obtener datos en tiempo real de todo lo que realiza la app a través de un navegador.

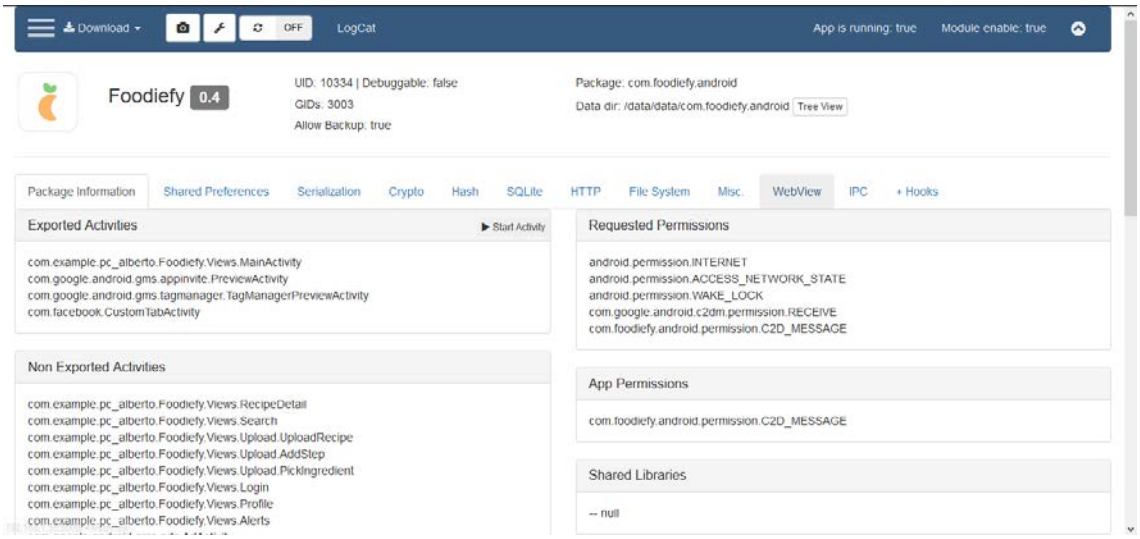


Ilustración 47 - Inspeckage mostrando algunos datos en tiempo real de la app

Con esta configuración se han podido verificar algunas peticiones HTTP que se realizan; principalmente imágenes, *token* y correo propio del usuario pero ninguna información de otros usuarios.

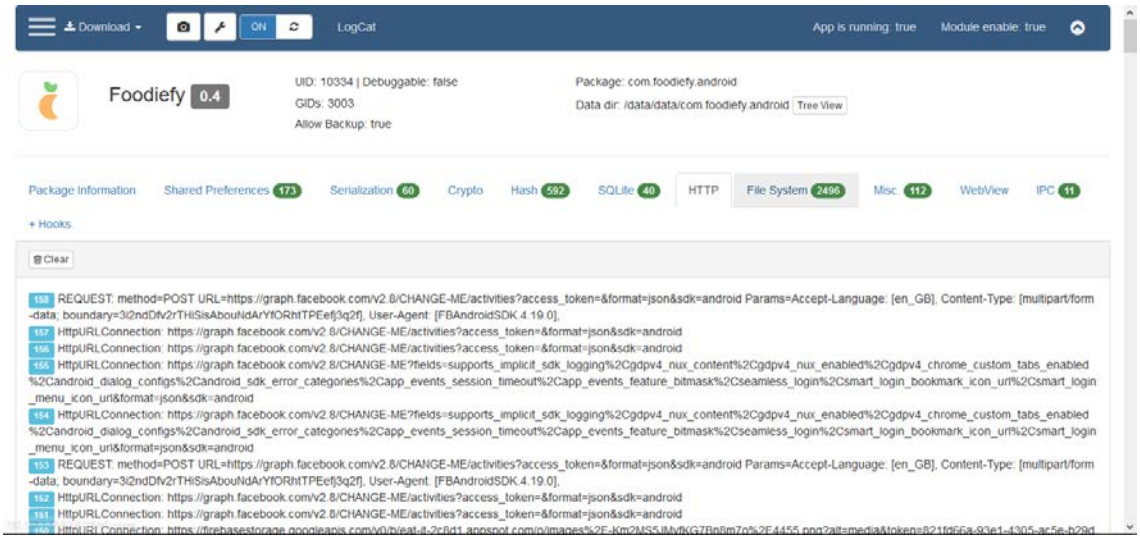


Ilustración 48 - Peticiones HTTP de la app

Al acceder a la pestaña "*Hash*" se observa porqué no es posible obtener datos. Parece que todos los datos se envían cifrados mediante SHA-1[27] y únicamente se envían los necesarios tal y como se muestra en las siguientes imágenes.

[27] "Secure Hash Algorithm – 1" es una función diseñada por la NSA americana y usada en criptografía.

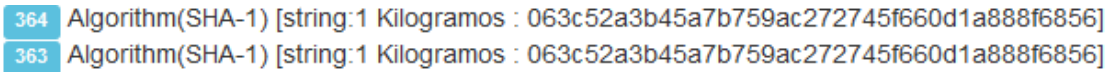


Ilustración 49 - Información cifrada con SHA-1 (1)

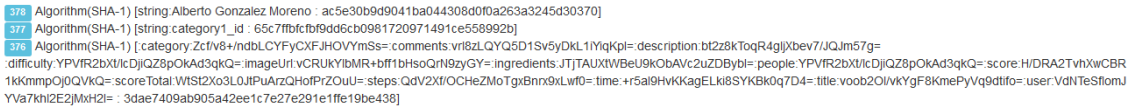


Ilustración 50 - Información cifrada con SHA-1 (2)

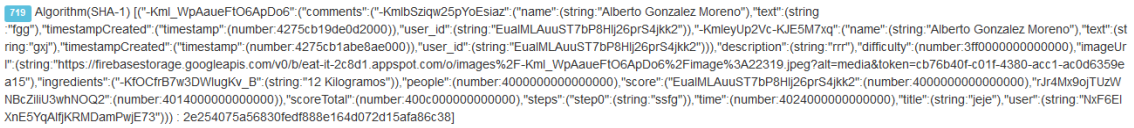


Ilustración 51 - Información cifrada con SHA-1 (3)

Por último, se ha de-compilado el APK [28]generado con Android Studio para intentar acceder a la información almacenada. Apenas se ha encontrado información debido a que se encuentra también codificada gracias al proceso de *minificación* y ofuscación.

7.1.1. Conclusión

Tras una inspección con las herramientas mencionadas no ha sido posible encontrar ninguna vulnerabilidad que permitiera acceder a los datos de otros usuarios, modificar datos, realizar acciones no permitidas…

La app, junto a la base de datos de Firebase, parece ser segura a nivel de peticiones HTTP y tráfico de paquetes.

[28] "Android Package Kit" es el formato de archivos que usa Android para la distribución e instalación de apps. Contiene todo el código y recursos usado por la aplicación, "empaquetado" todo en un archivo.

8. Usabilidad

8.1. Personas

En términos de usabilidad [29]y UX, se llama “persona” a un documento que describe un usuario ficticio pero que bien podría ser real (arquetipo). Hay varias maneras de realizar estos documentos, de manera esquemática o descriptiva. El número de personas para un proyecto depende de la naturaleza del mismo.

Realizar estos documentos de personas aportan valor a los desarrolladores para que entiendan quiénes van a utilizar su diseño y de qué manera. Estos documentos son importantes de cara a usabilidad y experiencia de usuario porque no es lo mismo hacer un diseño para una persona muy metida en el ámbito tecnológico y con conocimientos técnicos, que otro diseño para una persona sin esos conocimientos.

En estos documentos se describe las necesidades y ‘preocupaciones’ que puede tener un usuario y de qué manera se solucionan con nuestra aplicación. Están redactados en un lenguaje propio del usuario, sin usar argot técnico.

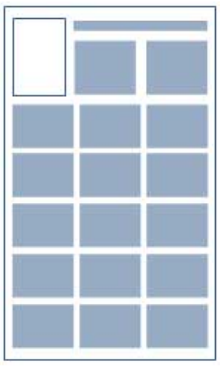
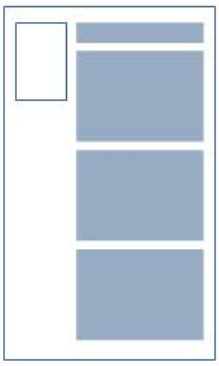
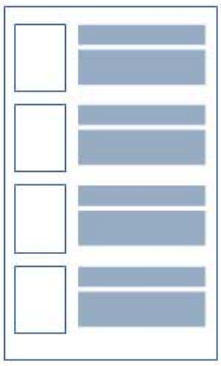
The Narrative	The Table	The Quick-and-Dirty
		
Best for stakeholders who are not so concerned about the technical details of user needs.	Best for designers who need an easy way to compare designs to user needs.	Best in situations where personas lack sufficient research.

Ilustración 52 - Formatos de documentos Personas. www.usability.gov

[29] Término que define a una aplicación o programa en medida que los usuarios son eficaces y eficientes cuando la usan.

8.1.1. Persona 1

Laura Fernández

foodiefy
DOCUMENTO PERSONA

EDAD 24
OCUPACIÓN Estudiante

Proactiva | Ocupada | Enérgica

"Me encantan los animales y por eso estudio veterinaria. Por las mañanas trabajo a media jornada en una escuela de adultos. ¡Así no hay quien tenga tiempo para nada!"

METAS

- Comer saludablemente
- Aprender algunos trucos de cocina
- Controlar la ingesta de nutrientes

FRUSTRACIONES

- Bajo nivel culinario
- Poco tiempo
- No tiene ayuda

CONOCIMIENTOS

Cocina

Aplicaciones móvil

Redes Sociales

BIO

Laura es una jóven estudiante que trabaja por las mañanas y estudia por la tarde. Vive sola en un piso de alquiler en Barcelona y tiene muy poco tiempo para pensar qué cocinar y hacer la comida. A Laura le gustaría poder saber casi inmediatamente qué puede cocinar y cuánto tiempo le llevaría sólo con abrir la nevera de su cocina. Lamentablemente sus conocimientos culinarios no son muy extensos.

8.1.2. Persona 2

Miguel González

foodiefy
DOCUMENTO PERSONA

EDAD 46
OCUPACIÓN Técnico de instalaciones

Cocinillas | Paciente | Saludable

"Me gustaría que hubiese una aplicación con un montón de recetas, centralizadas. Siempre cocino lo mismo y ya estoy cansado de buscar en libros de recetas."

METAS

- Encontrar nuevas recetas
- Compartir trucos culinarios

FRUSTRACIONES

- Siempre come lo mismo
- Buscar recetas en libros le resulta tedioso

CONOCIMIENTOS

Cocina

Aplicaciones móvil

Redes Sociales

BIO

Miguel es un adulto con conocimientos de cocina altos. Le gusta la cocina. Sus conocimientos de aplicaciones móviles son reducidos, aunque se defiende. A Miguel le gustaría poder variar de recetas ya que normalmente siempre come lo mismo, aunque saludable. Miguel debe vigiliar la ingesta de nutrientes por lo que siempre va con cuidado a la hora de buscar una receta. Es paciente y no tiene prisa a la hora de hacer su mejor plato.

8.1.3. Persona 3

Raúl del Río

foodiefy
DOCUMENTO PERSONA

EDAD 30

OCUPACIÓN Monitor de spinning

Saludable | Enérgico | Estricto

"Pavo, ternera, pollo y vuelta a empezar. ¡Qué rollazo!"

METAS

- Mantenerse en forma
- Encontrar nuevos platos

FRUSTRACIONES

- Tiene mucho papeleo con las dietas de la gente
- Siempre come lo mismo
- Arroz con pollo día y noche

CONOCIMIENTOS

Cocina

Aplicaciones móvil

Redes Sociales

BIO

Raúl es monitor de spinning en un gimnasio. Como la mayoría de gente fitness, debe cuidar mucho su dieta. Come siempre los dos mismos grupos de alimentos y ello le frustra. Además, Raúl controla la dieta de algunos de sus chicos/chicas en el gimnasio.

8.2. Prueba de usabilidad

En una fase muy temprana del desarrollo se ha realizado una **primera** prueba de usabilidad.

Para esta primera prueba se reclutaron a diez estudiantes del CITM-UPC y se realizó el test en el Laboratorio de Interacción Humano-Computadora disponible en las instalaciones del centro.

En el anexo de este documento se pueden encontrar tanto el documento de planificación del test (Anexo 2), como el documento de resultados (Anexo 3), ambos documentos en inglés.

Casi consecutivamente a esta primera prueba realizada en las instalaciones del CITM a modo de prueba, repetimos el test a más gente con la diferencia de que ahora ya no nos encontramos en un laboratorio y los usuarios no son estudiantes de un centro educativo relacionado con las tecnologías. Para este test no se ha preparado un documento de plan de test puesto que es lo mismo que en el anterior, a excepción de los usuarios y el lugar.

Con los resultados del test concluimos que hay cuatro mejoras posibles en la aplicación.

- Destacar el icono de subir receta en la barra de menú inferior
- Dar información al usuario de cómo debe introducir el criterio de búsqueda cuando se busca una receta y dar la opción de elegir el método de búsqueda (recetas por ingrediente, recetas por campo de texto libre, otros usuarios, categorías…)
- El icono de notificaciones no se percibe como notificaciones, sino como usuarios/amigos/perfil.
- Aclarar en la pantalla de subir receta la sección de descripción y procedimiento

Para más información detallada consultar los anexos.

Dada la información extraída de los resultados del test, la aplicación requiere que se le añada:

- Una pantalla previa para escoger el método de búsqueda
- Una clase Java para lidiar con las unidades de medidas en los ingredientes, que sea estándar y con la posibilidad de cambiar entre unidades.

9. Planificación del proyecto y presupuesto

- Definir el problema – 1 semana (Paquete de trabajo 1: **Gestión y coordinación**)
- Definir imagen corporativa – 2 semanas (Paquete de trabajo 2: **Diseño**)
- Análisis de competencias y productos similares – 1 semana (Paquete de trabajo 2: **Diseño**)
- Definir plan de marketing – 3 semanas (Paquete de trabajo 4: **Promoción**)
- Definir diseño de la app – 4 semanas (Paquete de trabajo 2: **Diseño**)
- Definir estructura interna de la app – 2 semanas (Paquete de trabajo 3: **Desarrollo**)
- Programar la app – 3 meses (Paquete de trabajo 3: **Desarrollo**)

Teniendo en cuenta la planificación anterior y que el proyecto se lleva a cabo en 5 meses de trabajo:

	Categorías de gasto					Fondos		
	Costes de personal directos	Otros costes directos	Costes de subcontratación	Costes indirectos	Total eligible costs	Ratio de reembolso	Importe máximo de subvención	Importe pedido para subvención
Partner 1 - Roger	3.850	265	0	1.029	5.144	100%	5.144	5.144
Partner 2 - Alberto	3.850	245	0	1.024	5.119	100%	5.119	5.119
Total asociados	7.700	510	0	2.053	10.263	2	10.263	10.263

Tabla 3 - Resumen de presupuesto

	Partner	Short name	Type	Country	Average weekly cost (€)	Funding rate						
	1	Roger	Profit	Spain	350	100%						
	WP1	WP2	WP3	WP4	WP5	WP6	WP7	WP8	WP9	WP10	Total	
Personnel effort (time-units)	1	7	0	3							11	
Personnel costs	350	2.450	0	1.050	0	0	0	0	0	0	3.850	75%
Travel costs	0	20	0	20							40	1%
Equipment (amortisation only)	50	50	0	50							150	3%
Consumables	5	20	0	50							75	1%
Other direct costs	0	0	0	0							0	0%
Subcontracting	0	0	0	0							0	0%
Total direct costs	405	2.540	0	1.170	0	0	0				4.115	80%
Indirect costs	101	635	0	293	0	0	0				1.029	20%
Total budget (eligible costs)	506	3.175	0	1.463	0	0	0	0	0	0	**5.144**	100%
Requested EC funding	506	3.175	0	1.463	0	0	0	0	0	0	**5.144**	

Tabla 4 - Presupuesto para participante 1

	Partner	Short name	Type	Country	Average weekly cost (€)	Funding rate						
	2	Alberto	Profit	Spain	350	100%						
	WP1	WP2	WP3	WP4	WP5	WP6	WP7	WP8	WP9	WP10	Total	
Person-weeks per WP	1	0	10	0							11	
Personnel costs	350	0	3.500	0	0	0	0	0	0	0	3.850	75%
Travel costs	0	0	20	0							20	0%
Equipment (amortisation only)	75	0	75	0							150	3%
Consumables	5	0	20	0							25	0%
Other direct costs	0	0	50	0							50	1%
Subcontracting	0	0	0	0							0	0%
Total direct costs	430	0	3.665	0	0	0	0				4.095	80%
Indirect costs	108	0	916	0	0	0	0				1.024	20%
Total budget (eligible costs)	538	0	4.581	0	0	0	0	0	0	0	**5.119**	100%
Requested EC funding	538	0	4.581	0	0	0	0	0	0	0	**5.119**	

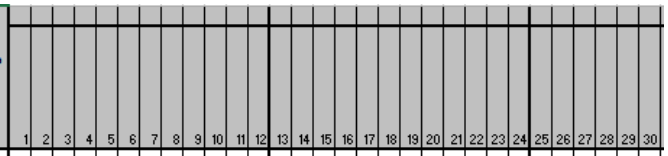
Tabla 5 - Presupuesto para participante 2

Recursos necesarios:

- Precio del diseñador | 6 euros/hora
    - Definir la imagen corporativa: Duración 2 semanas – 168 euros
    - Definir el diseño de la *app*: Duración 4 semanas – 336 euros
    - Grafismos necesarios para acciones de marketing: Duración promedia 3 horas a 20 euros/h cada grafismo
    - Diseñar la web: Duración 2 días – 100 euros
- Precio del programador | 6 euros/hora
    - Definir estructura interna de la app: Duración 2 semanas – 168 euros
    - Programar la app: Duración 3 meses – 1017 euros
- Precio de marketing manager | 6 euros/hora
    - Análisis de competencias y productos: Duración 1 semana – 84 euros
    - Plan de marketing: Duración 3 semanas – 252 euros
- Precio del dominio y servidor web | 12 euros/año
- Precio de la cuenta de desarrollador en Google Play | 25 euros
- Precio de la base de datos | Gratuita
- Precio de las máquinas | 800-1200 euros/ordenador – 1600-2400 euros
- Precio de las licencias | Adobe Creative Cloud 60 euros / mes – 240-336 euros

9.1. Diagrama de Gantt

ID	TITLE	Duration (in weeks)	Starting week	Ending week
		107		
WP1	Management	30	1	30
T1.1	Documentación sobre hábitos de consumo	3	1	3
T1.2	Documentación sobre Android y Firebase	3	4	7
T1.3	Crear proyecto Android	1	8	9
T1.4	Crear cuenta GitHub	1	9	10
T1.5	Añadir proyecto al repositorio de Git	1	10	11
T1.6	Implementar Git	1	11	12
T1.7	Definir estructura de la BBDD	2	12	14
T1.8	Conectar Firebase con el proyecto	1	14	15
T1.9	Lógica pantalla MainActivity	1	14	15
T1.10	Lógica pantalla RecipeDetailActivity	1	15	16
T1.11	Lógica pantalla SearchActivity	1	16	17
T1.12	Lógica pantalla UploadActivity	1	17	18
T1.13	Analisis de usabilidad	2	19	21
T1.14	Firebase Storage	1	21	22
T1.15	Authentication	1	22	23
T1.16	Lógica panalla LoginActivity	1	23	24
T1.17	Menus	1	24	25
T1.18	Navegación	1	25	26
T1.19	Lógica pantalla ProfileActivity	1	26	27
T1.20	Recibir notificaciones desde la consola	1	27	28
T1.21	Enviar notificaciones automáticas	1	28	29
T1.22	Layouts	6	22	28
T1.23	Firebase Cloud Functions	2	29	31
T1.24	Lógica pantalla AlertsActivity	1	30	31



En este diagrama de Gantt está la planificación por semanas de cada tarea a desarrollar. Aunque la duración total son 30 semanas, se descuentan las 8 primeras semanas dedicadas a documentación y plantear el desarrollo del proyecto. Por lo tanto, en realidad, son 22 semanas.

9.2. Competencia

Actualmente el principal competidor en el mercado es Hatcook. Es una *startup* [30]que empezó en 2009 y publicando su app en la App Store en 2011 llegando a ser la número 1 en España.

Durante el 2015 se convirtió en una red social y en 2016 superaron los 2 millones de descargas, siendo una de las apps de cocina más destacadas en España y Latinoamérica.

Hatcook está financiada por el programa **bStartup 10** del **Banco Sabadell**, por **ENISA** (sociedad mercantil estatal, dependiente del Ministerio de Economía, Industria y Competitividad, a través de la Dirección General de Industria y de la Pequeña y Mediana Empresa) y por **ADE2020** (Aceleradora de Empresas de la Junta de Castilla y León).

Pese a la presencia de una competencia fuerte, no hay que olvidar que Foodiefy podría superar en todos los sentidos a Hatcook teniendo mucho menos presupuesto y apoyo.

[30] Empresas que buscan arrancar, emprender o montar un nuevo negocio, y aluden a ideas de negocios que están empezando o están en construcción, y generalmente se trata de empresas emergentes apoyadas en la tecnología. www.wikipedia.org

10. Metodología y rigor

Para planificar las tareas y tener una visión global del proyecto, se ha usado la aplicación Trello. En cuanto a la comunicación interna entre los integrantes del proyecto se ha solventado con WhatsApp, Skype y comunicación presencial principalmente. Para compartir los archivos se ha hecho uso de Drive. Se ha utilizado un sistema de control de versiones con Git, alojando el proyecto en GitHub y así tener acceso desde cualquier ordenador a la versión más actualizada.

10.1. Agile - Scrum

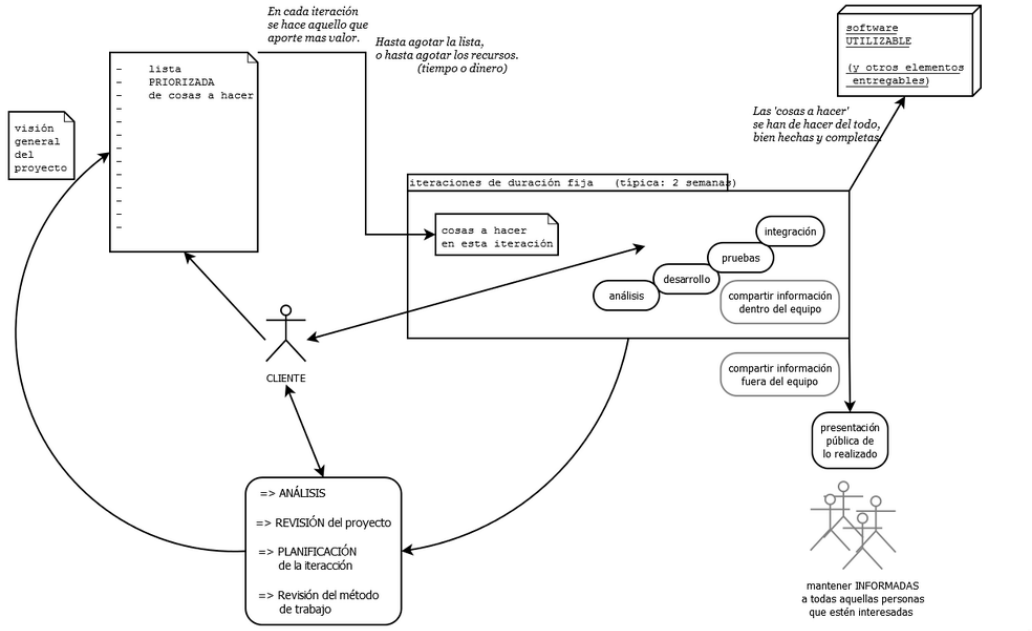


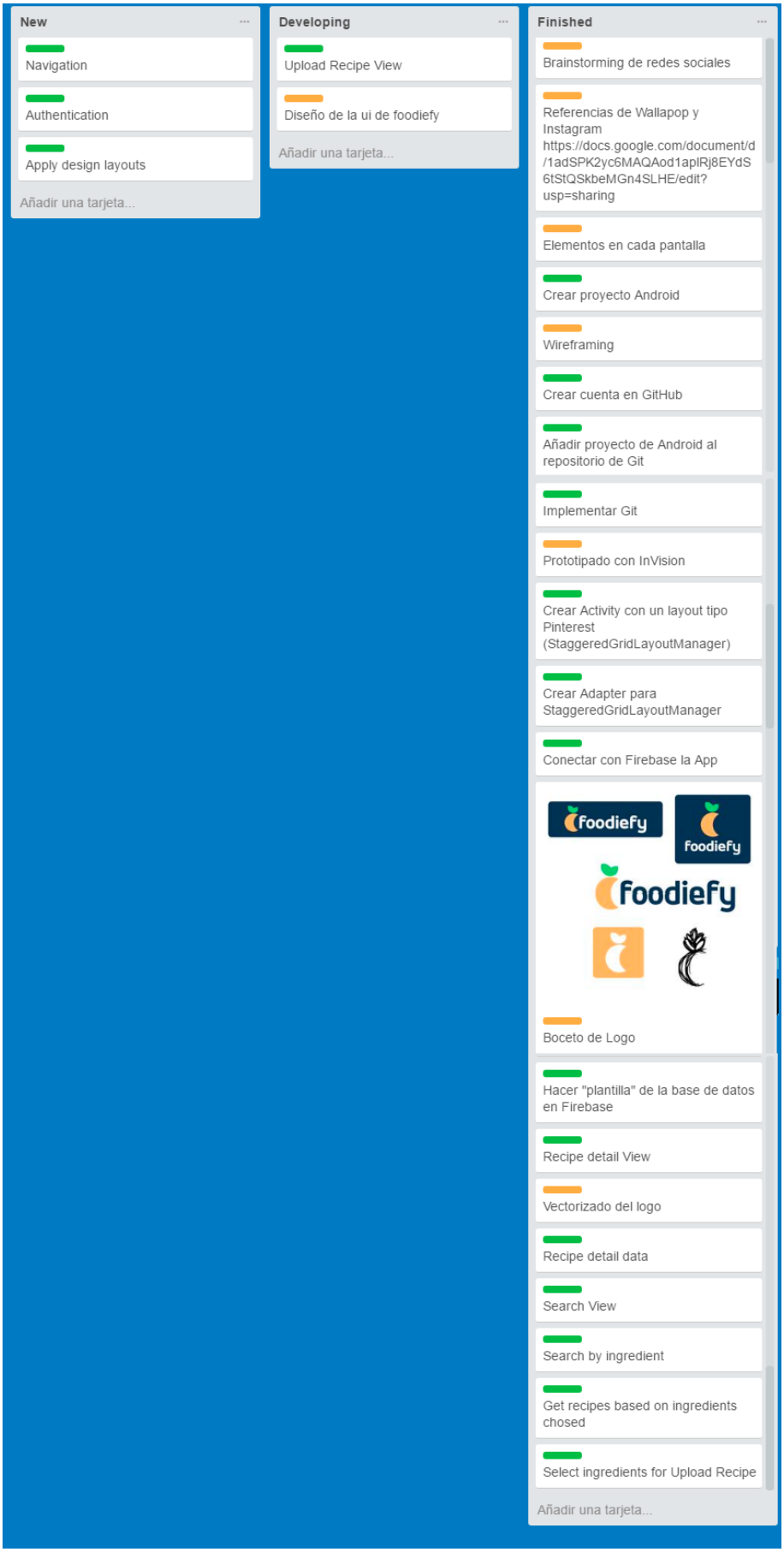
Ilustración 53 - Esquema general de una metodología ágil para desarrollo de software. www.wikipedia.org

Durante el desarrollo de la aplicación, el equipo formado por dos personas, se llevará a cabo una metodología ágil. En esta metodología se hacen iteraciones en las cuales se modifican los requisitos y soluciones dependiendo de las necesidades del proyecto. De esta manera, en una iteración puede haber las tareas A, B y C para hacer y luego en la siguiente iteración aparece una tarea D con más prioridad.

A cada iteración se le llama sprint y en este caso dura dos semanas cada sprint. Al final de cada sprint se reúnen los dos desarrolladores e intercambian opiniones y se ponen al día (reunión presencial) aunque durante el sprint siguen en contacto mediante Skype y WhatsApp. Durante esta reunión bisemanal, se valida que lo que se está haciendo sea lo que el proyecto requiere y que no haya habido ningún malentendido.

Como se ha mencionado, para la gestión de las tareas y tener una visión global del desarrollo, se usa Trello.

10.1.1. Trello


New
Navigation
Authentication
Apply design layouts
Añadir una tarjeta...
Developing
Upload Recipe View
Diseño de la ui de foodiefy
Añadir una tarjeta...
Finished
Brainstorming de redes sociales
Referencias de Wallapop y Instagram https://docs.google.com/document/d/1adSPK2yc6MAQAod1aplRj8EYdS6tStQSkbeMGn4SLHE/edit?usp=sharing
Elementos en cada pantalla
Crear proyecto Android
Wireframing
Crear cuenta en GitHub
Añadir proyecto de Android al repositorio de Git
Implementar Git
Prototipado con InVision
Crear Activity con un layout tipo Pinterest (StaggeredGridLayoutManager)
Crear Adapter para StaggeredGridLayoutManager
Conectar con Firebase la App
foodiefy
foodiefy
foodiefy
Boceto de Logo
Hacer "plantilla" de la base de datos en Firebase
Recipe detail View
Vectorizado del logo
Recipe detail data
Search View
Search by ingredient
Get recipes based on ingredients chosed
Select ingredients for Upload Recipe
Añadir una tarjeta...

11. Plan de marketing

11.1. Plan de marketing del TFG

11.1.1. Descripción de la situación

Actualmente en la sociedad las aplicaciones más descargadas y con mayor tráfico de usuarios son las aplicaciones móviles de redes sociales como pueden ser Whatsapp, Facebook, Twitter etc. En una de las gráficas se ve que del total de horas medias que los españoles usan el *Smartphone*, que concretamente es de 1h y 55 min, solamente 19 minutos son de media los que no están en una red social.

A nivel mundial, los españoles son de las poblaciones que más usan los *Smartphone* y los que más consumo hacen de internet.

Con estos números, se entiende que el mercado está presente y disponible. También hay posibilidades de encontrar una necesidad que tengan los usuarios y así solventarla.

Ilustración 54 - Infográfico de We Are Social

Entorno a la comunidad y tráfico de usuarios que ven contenido de cocina por internet, nuestro potencial público está alojado en *Youtube*. Esto se debe a que esta aplicación es una canalización de este contenido a través de un medio distinto y con un uso distinto, pero con un mismo público potencial.

Cocina para todos: https://www.youtube.com/user/cocinatodo

Con más de un millón de suscriptores es el canal de cocina que más público tiene. Con una media de 200.000 reproducciones en sus videos y videos con máximos de 9.000.000 de reproducciones.

Existen otros canales no tan famosos, pero que crecen poco a poco:

Lolita la pastelera: https://www.youtube.com/user/lolitalapastelera

Las recetas de MJ: https://www.youtube.com/user/lasrecetasdemj

Tuiwok estilo: https://www.youtube.com/user/tuiwokestilo

El cocinero fiel: https://www.youtube.com/user/elcocinerofiel

Javi recetas: https://www.youtube.com/user/javirecetas

11.1.2. Análisis de la situación - DAFO

Ilustración 55 - Análisis DAFO

11.1.3. Fijación de objetivos

El principal objetivo para Foodiefy es darse a conocer como la *app* de recetas de referencia, pudiendo crear una comunidad estable de usuarios que agrande la aplicación. Sin comunidad, la *app* no es nada. Por ello el objetivo principal tras 1 año de campaña y una vez publicada la aplicación es conseguir una comunidad de usuarios sólida y estable y que la propia *app* se retroalimente sola.

Más a largo plazo el siguiente objetivo a nivel de marca es que la gente reconozca la funcionalidad de la app con ver el logotipo. Si se piensa en WhatsApp, se viene a la mente la función de enviar mensajes, o en Instagram en ver fotos. Para Foodiefy, se quiere que al ver el logo se piense automáticamente en búsqueda de recetas.

A corto plazo se ha marcado como objetivo que la aplicación tenga un tráfico notable y las personas se interesen por ella. También que los usuarios puedan subir sus propias recetas, puntuarlas y comentarlas.

Como en toda red social, es tendencia que haya cierta gente que se haga popular y destaque por encima de los demás usuarios en términos de impresiones, *likes*, comentarios, puntuaciones positiva... A corto plazo se quiere poder ir viendo indicios de algunos usuarios que puedan ir tendiendo a estos niveles.

11.1.4. Estrategias

Para llegar a cumplir los objetivos marcados tanto a largo como a corto plazo, se ha pensado en cuatro etapas en las que se va a resumir esta estrategia de marketing:

Etapa 1: Se busca enseñar a la gente qué hace exactamente la *app* y qué problemas puede llegar a solucionar. Para ello se necesita crear esa necesidad que en muchos casos ya existe, pero en otros por pereza a pensar, no.

Etapa 2: Se busca que los usuarios con esa necesidad descarguen la *app*, que empiecen a indagar qué se puede hacer con ella y cómo le pueden sacar el mayor partido.

Etapa 3: Cuando ya haya usuarios que están descargándose la *app* para ver qué hace y cómo la pueden usar y ver si les conviene o no, se requiere fidelizar a estos usuarios para llegar a crear una comunidad estable. Una red social sin gente que comente, que puntúe o que suba contenido no es gran cosa.

Etapa 4: Retener a los usuarios, se requiere un uso continuo de la *app.*

11.1.5. Plan de acción

En relación a las cuatro etapas mencionadas, se definen qué acciones se realizarían en cada una de ellas para llegar a los objetivos especificados.

Etapa 1 y 2 – Atracción de público: Cerrar acuerdos con *youtubers* e *instagramers* del mundo de la cocina. Algunos de ellos ya se han mencionado con anterioridad y manejan unos niveles de impresiones grandes, pudiendo llegar a millones de personas en un par de videos.

También el trabajo y constancia en redes sociales va a ser un factor determinante. Aunque Foodiefy sea una red social, se requiere buscar el público en redes sociales más estándares como pueden ser Twitter y Facebook. Para ello se irán lanzando una serie de anuncios en forma de publicación sobre algunas de las funcionalidades y problemas cotidianos que resuelve la *app* diariamente. Además, la intención es realizar

un vídeo estilo infográfico en movimiento explicando todo lo que ofrece la *app*. Este vídeo será publicado en las redes sociales y en los distintos acuerdos con *influencers* del mundo de la cocina en internet.

Etapa 3 - Fidelización: Poner en la *app* una especie de mini guía para aquellos nuevos usuarios que se la hayan descargado para que les explique todo lo que ofrece. No se puede permitir que haya usuarios que la descarguen y la borren al momento sin haber visto todas las posibilidades que ofrece, o por lo menos que ese porcentaje de usuarios sea mínimo.

Para esta etapa, ese sería el trabajo. Lo siguiente sería que a la gente le guste y se vea totalmente a gusto usándola. Para cualquier duda de funcionamiento se debe poder ofrecer a los usuarios una forma de contacto o de ayuda. Se busca evitar la pérdida de usuarios por descontento o enfado. Para ello se atenderá a través de nuestras redes sociales como son Twitter y Facebook todas las peticiones que se hagan, además de que la propia *app* tendrá una sección de ayuda con preguntas frecuentes.

Etapa 4 – Retención: Wallapop trabaja muy bien este apartado y por ello se fija como referente. Cuando detecta que los usuarios están dejando de usar la aplicación, envía una especie de notificación recordando que se pueden hacer muchísimas cosas con la app y que se puede ganar dinero vendiendo cualquier cosa que haya por casa. Para Foodiefy se busca hacer lo mismo.

11.2. Plan de marketing personal

¿Cansado de no saber qué comer? ¿No tienes ganas de esperar al repartidor? **Foodiefy** te trae la solución. Foodiefy es una app de recetas de cocina que funciona a modo de red social. Y tal vez te preguntas "¿Qué la diferencia de otras apps de recetas?". La respuesta es fácil: **eficacia**, **eficiencia** y **sencillez**. Resultados precisos, de manera rápida y fácil.

"¿Y ya está?". No, todavía no. Foodiefy es una aplicación **nativa** con la que bien puedes ser el mejor chef o bien puedes ser un *manazas* en la cocina. No importa, tienes un hueco en esta aplicación. En el primer caso, seguramente te interese compartir tus mejores trucos y recetas con la comunidad. En el segundo caso puede que tengas tres ingredientes en la nevera y no saber cómo combinarlos: **Foodiefy**.

"La verdad, me esperaba algo más…". Bien, ¿qué me dirías entonces si te ofrezco información detallada sobre los valores nutricionales? De esta manera podrías controlar qué comes, y si vamos un poquito más allá, ¿por qué no dejar que controle tu dieta alguien especializado? Así es, con Foodiefy todo esto es posible.

"Wow, esto tiene mejor pinta. Pero… ¿mi madre podrá usarla?". ¡Por supuesto! De nuevo: **sencillez**. Se ha diseñado esta app para que cualquier persona pueda mostrar sus conocimientos culinarios al mundo, o comentar la receta de tu vecino.

Todo lo mencionado ya es posible con Foodiefy. Mi nombre es Alberto y soy el desarrollador de Foodiefy.

A título personal, promocionamos nuestra *app* en nuestras cuentas de Facebook, Twitter y LinkedIn. En la página web de la aplicación se especifica quién ha hecho la aplicación y con qué objeto. También dentro de la propia aplicación deberá haber un apartado "Sobre nosotros" que referenciará a la página web. En la propia página de la *app* en Google Play también aparecen los desarrolladores.

De esta manera la *app* queda totalmente "firmada" por nosotros como los desarrolladores.

12. Desarrollo

12.1. Estructura del proyecto

- /app/src/main/java/*package_name*/
  - **Adapters**
    - Alerts
      - AlertsAdapter.java
      - AlertViewHolder.java
      - ItemTouchHelperAdapter.java
      - TouchHelper.java
    - Comments
      - CommentsAdapter.java
      - CommentViewHolder.java
    - RecipeList
      - RecipeListAdapter.java
      - RecipeHolder.java
    - Search
      - RecyclerViewSearchAdapter.java
      - SearchViewHolder.java
      - TagListAdapter.java
    - Upload
      - StepsAdapter.java
      - StepsViewHolder.java
  - **Models**
    - Alert.java
    - Comment.java
    - Difficulty.java
    - Ingredient.java
    - Recipe.java
    - User.java
  - **Services**
    - MyFirebaseInstanceIDService.java
    - MyFirebaseMessagingService.java
  - **Utils**
    - CircleTransform.java
    - MapUtils.java
    - ToolbarLogoIcon.java
  - **Views**
    - Upload
      - AddStep.java
      - PickIngredient.java
      - UploadRecipe.java
    - Alerts.java
    - Login.java
    - MainActivity.java
    - Profile.java
    - RecipeDetail.java
    - Search.java

La estructura del proyecto se distribuye en Views, Utils, Models, Adapters y Services. Esta es una estructura típica.

Las Views son los distintos elementos de la interfaz gráfica de Android, es decir; las diferentes “pantallas” de la aplicación o *Activities*.

Se le llama Utils a aquellas clases Java que son utilidades, que no pertenecen a nada en concreto y que realizan una acción determinada.

Los Modelos son aquellas clases Java que se instancian a lo largo de la aplicación. Por ejemplo la clase Recipe, que tiene todos los atributos y métodos necesarios. Para integrar la aplicación con Firebase y que la descarga de datos sea un poco más eficiente y automática, la clase Java debe de coincidir con la estructura en Firebase.

Por último, los Adaptadores se encargan de hacer de puente entre el centro de datos y la aplicación (gestionan los datos y proporcionan una Vista individual por cada “registro” de datos, si fuera una tabla de datos, cada fila tendría una Vista).

Los Servicios son unas clases que ejecutan código en segundo plano y están constantemente activas. En este caso, por ejemplo, un Servicio se encarga de recoger las notificaciones y mostrarlas mientras que el otro Servicio está atento de si el *token* del usuario cambia (identificador del dispositivo, si me conecto desde otro teléfono mi *token* cambiará), y así poder enviar notificaciones al nuevo dispositivo (las notificaciones se envían a dispositivos según su *token*).

Todo lo mencionado se encuentra en el directorio /app/src/main/java/*package_name*/. Por otra parte, los recursos se guardan en otro directorio, concretamente en /app/src/main/res/.

- /app/src/main/res/
  - **drawable**
    - ic_vector_logo_horizontal.xml
    - …
  - **layout**
    - activity_alerts.xml
    - alert_item.xml
    - …
  - **menu**
    - bottom_navigation_main.xml
    - menu_main.xml
  - **mipmap**
    - image.png
      - image.png (hdpi, mdpi, xhdpi, xxhdpi, xxxhdpi)
  - **values**
    - colors.xml
    - dimens.xml
    - strings.xml
    - styles.xml

Para la representación simplificada de la estructura de recursos, solo se muestran ejemplos.

Los archivos *drawable* pueden ser de tipo mapa de bits (jpeg, png, etc) aunque lo recomendado es que únicamente sean vectores. En el caso de Android, estos archivos vectoriales se tratan como archivos *xml* mientras que los mapas de bits es recomendable ponerlos en *mipmap*. Android Studio automáticamente crea diferentes versiones del mismo mapa de bits con distintas resoluciones (hdpi, mdpi, xhdpi, xxhdpi y xxxhdpi) y Android decide qué versión coger.

Por otra parte la carpeta de *layouts* contiene archivos *xml* [31]que describen las vistas que se encuentran en cada Activity y cómo se muestran (activity_alerts.xml). También pueden describir vistas individuales (alert_item.xml).

La carpeta *menu* es opcional ya que contiene archivos *xml* que podrían estar en *layout*, pero para mantener una mejor organización y accesibilidad a los recursos se han separado a otra carpeta. Estos archivos *xml* son como los descritos.

Finalmente, en la carpeta *values* se definen varios archivos *xml* que contienen valores distintos que se usan en la aplicación. Estos valores pueden ser como un código de color, un número entero, una lista de cadenas de texto… etcétera. Por ejemplo, en el archivo *colors.xml* se define:

```
<color name="colorPrimary">#273854</color>
```

Para el color “colorPrimary” se define el valor “#273854”.

De esta manera, es posible definir valores, usarlos en distintas partes de la app y más tarde cambiarlos todos modificando directamente el archivo correspondiente de *values*. Así se evitan errores humanos, se ahorra tiempo y se mejora el código al no tener *hardcodedado*[32] los valores. Adicionalmente, se pueden definir valores específicos para

[31] *XML*, siglas en inglés de eXtensible Markup Language, traducido como "Lenguaje de Marcado Extensible" o "Lenguaje de Marcas Extensible", es un meta-lenguaje que permite definir lenguajes de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible. www.wikipedia.org

[32] *Hard-code*, término del mundo de la informática hace referencia a una mala práctica en el desarrollo de software que consiste en incrustar datos directamente (a fuego) en el código fuente del programa, en lugar de obtener esos datos de una fuente externa

idiomas concretos. De tal manera que se puede definir un texto con nombre “myText” y que Android coja la traducción correspondiente:

- res/values
  - strings.xml
- res/values-en
  - strings.xml

Ambos archivos *strings.xml* contienen una definición para un texto “myText”, pero con un valor distinto. Android buscará la carpeta correspondiente de *values* según el código de idioma y si no la encuentra lo buscará en la carpeta por defecto (*values* sin código de idioma).

12.2. Arquitectura de la app

A continuación se exponen siete gráficos. El primero corresponde a una “vista aérea” del diagrama *UML* [33]de la aplicación, y seguidamente los seis otros gráficos corresponden a diferentes partes de la vista aérea repartidos según el Modelo que usen.

como un fichero de configuración o parámetros de la línea de comandos, o un archivo de recursos. www.wikipedia.org

[33] Por sus siglas en inglés, *Unified Modeling Language, e*s un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados. www.wikipedia.org

Ilustración 56 - Vista aérea del diagrama UML

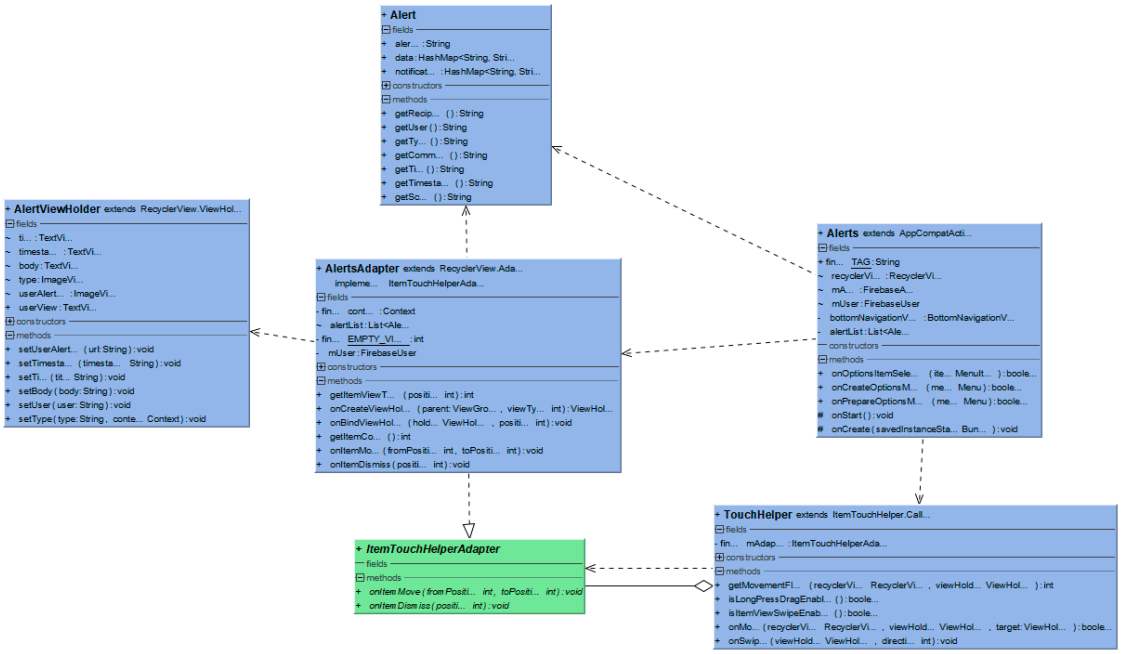


Ilustración 57 - Diagrama UML - Alert

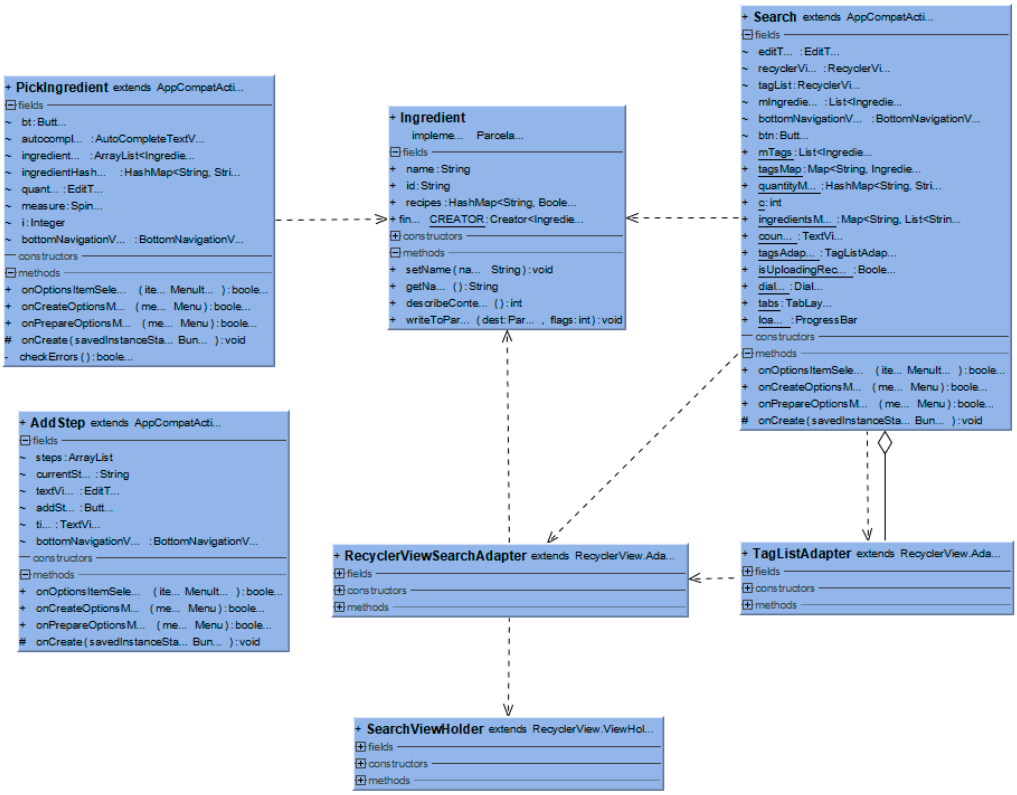


Ilustración 58 - Diagrama UML - Ingredient

Ilustración 59 - Diagrama UML - Recipe

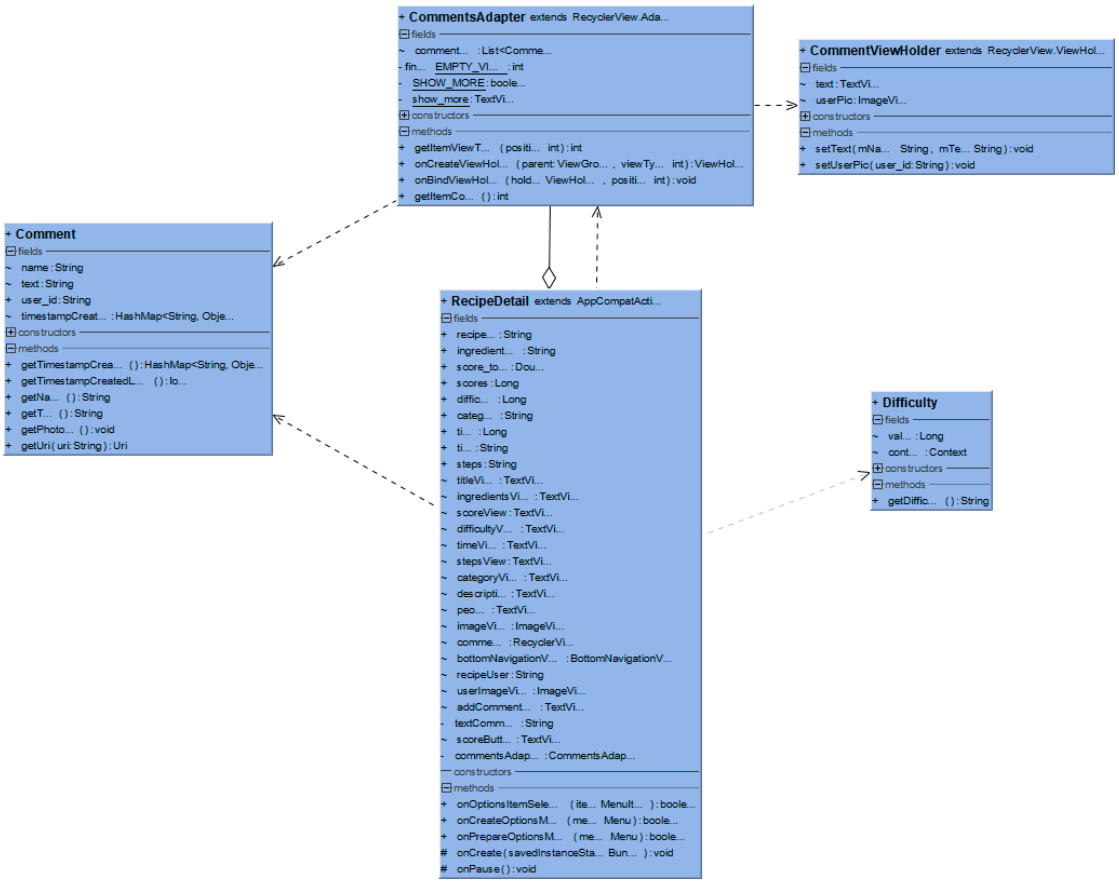


Ilustración 60 - Diagrama UML - Comment y Difficulty

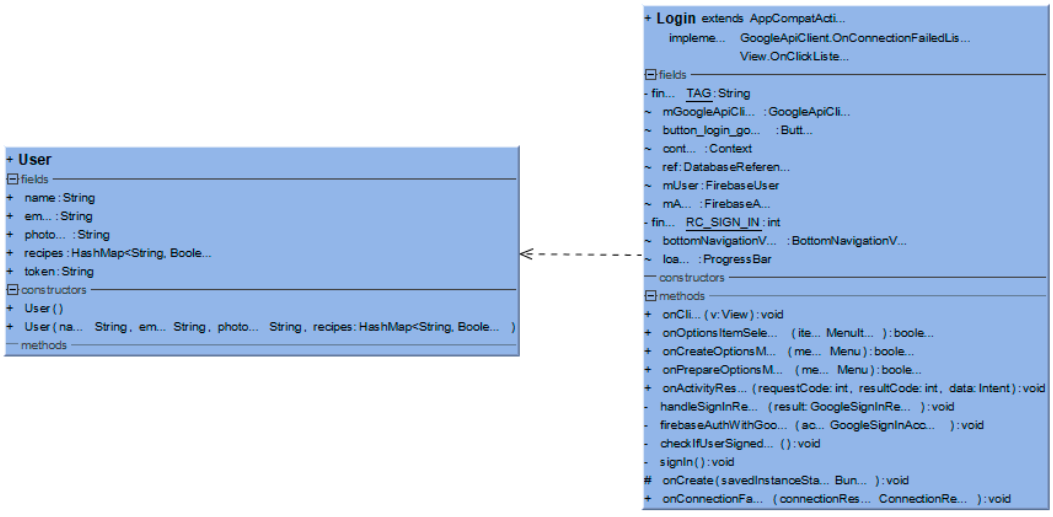


Ilustración 61 Diagrama UML - User

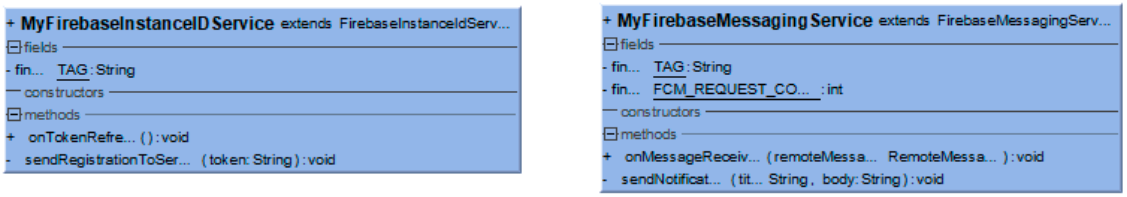


Ilustración 62 - Diagrama UML - Servicios

12.3. Programación de la app

En esta sección se expone a grandes rasgos los principales aspectos que conciernen a la programación de la app. Para ello se divide en tres bloques: las diferentes *Activity* o "pantallas" de la aplicación, los Servicios y *Layouts*.

12.3.1. Activities

Se han definido 9 clases, una para cada posible pantalla de la app, que son las siguientes:

- MainActivity: Esta es una pantalla en la que se muestra un listado de recetas. Actúa tanto de *home*, como de resultados de búsqueda. Es accesible desde la barra superior, tocando el logo de la app que aparece a la derecha.
- RecipeDetail: Cuando se toca una receta (*tap*, *click*) se abre esta pantalla conteniendo la información asociada a la receta. Desde aquí se puede ir al perfil de usuario que ha colgado la receta, comentarla y puntuarla.
- Search: Desde el menú principal (la barra inferior) se puede acceder a la pantalla de búsqueda al pulsar sobre la lupa. Esta pantalla contiene tres pestañas, una para cada método de búsqueda: recetas por ingredientes, recetas por campo de texto libre y perfiles de usuario. Este proyecto sólo aborda la primera y principal opción, mediante ingredientes. Las otras dos opciones quedan pendientes para una segunda versión de la app. Cuando se introducen criterios de búsqueda, la aplicación realiza esa búsqueda automáticamente en segundo plano y si se pulsa el botón de mostrar, abre la pantalla MainActivity mostrando las recetas específicas.
- UploadRecipe: Igual que la pantalla de búsqueda, desde el menú se puede acceder a la pantalla de subir una receta. Cuando se intenta acceder a esta parte de la aplicación, se comprueba si hay una sesión activa. Si no hay una sesión

activa, automáticamente se abre la pantalla Login para inciar sesión. Si por el contrario ya había una sesión activa, se abre la pantalla de subida.

- AddStep: Dentro de la pantalla de subida de receta, cuando se añaden pasos al procedimiento, se hace desde esta otra pantalla AddStep.Esta pantalla, junto a PickIngredient, no muestran el logo de la aplicación en la barra superior, sino que muestran una opción de “Cancelar”.
- PickIngredient: Igual que AddStep, pero para introducir ingredientes en la receta que se está subiendo.
- Alerts: Siguiendo el esquema de UploadRecipe, esta pantalla es accesible desde el menú inferior y se comprueba si existe una sesión activa. La pantalla Alerts muestra las notificaciones que ha recibido un usuario.
- Profile: Esta es la pantalla de perfil de usuario donde se muestran las estadísticas de usuario y las recetas que ha subido. También hay una opción de seguir al usuario. A esta pantalla se puede acceder desde una receta (pulsando sobre la foto del usuario) y se mostrará el perfil de ese usuario. También es accesible desde la pantalla de *home* (MainActivity) al pulsar sobre el icono de perfil que aparece en la barra superior a la derecha. Si hay una sesión activa, se mostrará el perfil del usuario actual. En cambio si no hay ninguna sesión activa se muestra la pantalla de inicio de sesión.
- Login: Login es la pantalla en la que el usuario puede iniciar sesión con su cuenta de Google. La única manera de acceder a esta pantalla es intentando entrar en Subir receta, Alertas o Perfil (el propio) sin una sesión activa.

Todas las Activities mencionadas son clases de Java que deben extender de la clase **Activity**. En este proyecto, concretamente extienden de la clase **AppCompatActivity** que ya hereda de **Activity**. Esto es necesario para hacer posible el uso del *Action Bar* (la barra superior en la que aparece el título y opciones) con la biblioteca de compatibilidad de Android (para versiones antiguas de Android).

Además de extender la clase AppCompatActivity, estas clases sobrescriben los siguientes métodos:

onCreate: Este método es el que se ejecuta cuando se lanza una Activity. Aquí se especifica el *layout* que se debe cargar, se inicializan valores, se inicializa la barra de menú superior, etcétera.

```
//Referencia de la barra superior (Toolbar)
Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
 //Referencia del botón del menu superior (hamburguesa)
 DrawerLayout drawerLayout = (DrawerLayout) findViewById(R.id.navigation_drawer);
  //Referencia de la lista del menu superior
  ListView mDrawerList = (ListView) findViewById(R.id.left_drawer);
   //Añadimos un listener a los items de la lista del menu superior
   mDrawerList.setOnItemClickListener(new DrawerItemClickListener());
    //Ponemos la toolbar como ActionBar
    setSupportActionBar(toolbar);
     final ActionBar actionBar = getSupportActionBar();
```

Ilustración 63 - Fragmento de código en el método onCreate donde se inicializa el menú superior.

onCreateOptionsMenu: Este método se ejecuta cuando se "crea" el menú superior (*ActionBar)*. En él se especifican los diferentes elementos que tiene la barra.

onPrepareOptionsMenu: Una vez acaba el método anterior, onCreateOptionsMenu, se lanza el método onPrepareOptionsMenu. En este método se debe programar qué hacen esos elementos de la barra superior cuando se les clicka. Por ejemplo, en la siguiente ilustración observamos que se especifica la visibilidad (*true* o *false*) de los ítems del menú con ID "cancel_action" y "myProfile". Concretamente, se está especificando que no se muestre la opción de cancelar y que en su lugar aparezca *myProfile*.

```
Override
ublic boolean onPrepareOptionsMenu(Menu menu)
{
    MenuItem cancel = menu.findItem(R.id.cancel_action);
    cancel.setVisible(false);


     MenuItem profile = menu.findItem(R.id.myProfile);
     profile.setVisible(true);

      final MenuItem item = menu.getItem(0);
      if(item.getItemId()== R.id.myProfile){
```

Ilustración 64 - Fragmento de código del método onPrepareOptionsMenu

onOptionsItemSelected: En este método se programa la acción a realizar cuando se selecciona una opción del menú. Por ejemplo:

```
erride
blic boolean onOptionsItemSelected(MenuItem item) {
  // handle arrow click here
  if (item.getItemId() == android.R.id.home) {
      finish();
  }

  if (item.getItemId() == R.id.cancel_action) {
      Intent intent = new Intent();
      setResult(RESULT_CANCELED, intent);
      finish();
  }

  return super.onOptionsItemSelected(item);
}
```

Ilustración 65 - Fragmento de código del método onOptionsItemSelected

En la ilustración anterior se definen las acciones a realizar cuando se pulsa sobre los ítems con ID “home” y “cancel_action”.

También definen una clase privada interna DrawerItemClickListener que implemente ListView.OnItemClickListener. Esta clase debe sobrescribir el método onItemClick y es la encargada de responder cuando se selecciona una opción del *menú de hamburguesa* (las tres rayitas) de la barra superior. Concretamente para este proyecto, únicamente gestiona la única opción que hay: salir de la sesión.

```
ate class DrawerItemClickListener implements ListView.OnItemClickListener {

@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id)
    switch (position){
        case 0:
            mAuth.signOut();
            break;
    }
}
}
```

Ilustración 66 - Implementación del menú superior.

Como se ve en la ilustración, la clase recibe un evento de *click* con la posición en el menú. Si esta posición es la número “0”, aplico el método signOut() sobre la instancia activa de FirebaseAuth.

A parte de todo lo mencionado, cada Activity implementa en su método onCreate la programación necesaria para la pantalla concreta.

12.3.2. Services

La aplicación tiene tan sólo dos servicios que son:

- MyFirebaseInstanceIDService: Esta clase Java extiende de la clase proporcionada por Firebase **FirebaseInstandeIdService**, que a su vez hereda de la clase **Service**. Este servicio es el encargado de gestionar los cambios del *token* de usuario. En otras palabras, cuando el identificador del dispositivo de un usuario cambia (puede ser porque ha abierto la aplicación en otro dispositivo o porque ha reinstalado la aplicación, etc) esta clase se encarga de notificar a la aplicación. Para ello se sobrescribe el método onTokenRefresh.

```
ic class MyFirebaseInstanceIDService extends FirebaseInstanceIdService {
private final static String TAG = "FirebaseInstanceID";
@Override
public void onTokenRefresh() {
    //super.onTokenRefresh();
    String token = FirebaseInstanceId.getInstance().getToken();
    Log.d(TAG, "Token has been refreshed - " + token);

    sendRegistrationToServer(token);
}
```

Ilustración 67 - Fragmento de código del Servicio FirebaseInstanceIdService

En el código de la ilustración tan solo se llama al método sendRegistrationToServer pasándole como parámetro el nuevo *token* generado. Este método se encarga de guardar el identificador en la base de datos para poder enviar notificaciones a dispositivos concretos.

- MyFirebaseMessagingService: Igual que el Servicio anterior, Firebase proporciona una clase base (**FirebaseMessagingService**) que extiende de **Service**. Este Servicio extiende de la clase proporcionada por Firebase. Este Servicio es el encargado de mostrar la notificación cuando el dispositivo la recibe. Para ello, únicamente hay que sobrescribir el método onMessageReceived.

```
class MyFirebaseMessagingService extends FirebaseMessagingService {
ivate static final String TAG = "FCM Service";
ivate static final int FCM_REQUEST_CODE = 000;

@Override
public void onMessageReceived(RemoteMessage remoteMessage) {

    if(remoteMessage.getNotification() != null){
        sendNotification(remoteMessage.getNotification().getTitle(), remoteMessage.ge
    }
}
```

Ilustración 68 - Fragmento de código del Servicio FirebaseMessagingService

En la ilustración se observa que cuando se recibe un mensaje, se llama al método sendNotification pasándole como parámetros el título y el contenido de la notificación. El método sendNotification es el encargado de crear la notificación de Android y mostrarla.

```
private void sendNotification(String title, String body) {
    Intent intent = new Intent(getApplicationContext(), Main
    intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);

    PendingIntent pendingIntent = PendingIntent.getActivity

    Uri notificationSound = RingtoneManager.getDefaultUri(
    NotificationCompat.Builder notificationBuilder = new
            .setContentTitle(title).setContentText(body)
            .setContentIntent(pendingIntent);
    NotificationManager notificationManager = (Notificat
```

Ilustración 69 - Fragmento de código del método encargado de mostrar notificaciones push en el Servicio FirebaesMessagingService

12.3.3. Layouts

En el proyecto existen 32 *layouts* distintos entre los cuales hay definidos *layouts* para *Activities* y para Vistas específicas.

Por ejemplo:

- activity_main.xml
- recipe_card.xml

El primer *layout* define la disposición en pantalla de los elementos para la Actividad MainActivity. Esta MainActivity contiene un *RecyclerView* que se rellena con Vistas individuales proporcionadas por el Adaptador. En este caso concreto, el Adaptador RecipeListAdapter.java proporcionará una Vista por cada receta. A cada receta se le indica que use el *layout* recipe_card.xml.

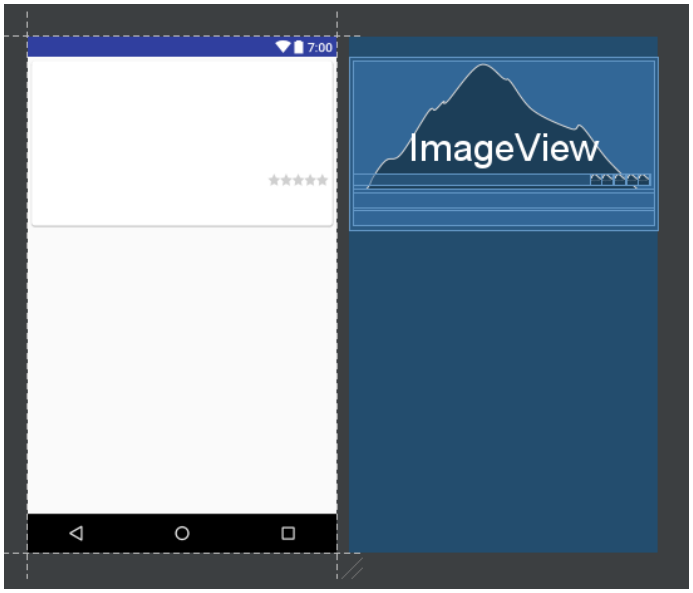


Ilustración 70 - Vista previa del *layout* recipe_card.xml

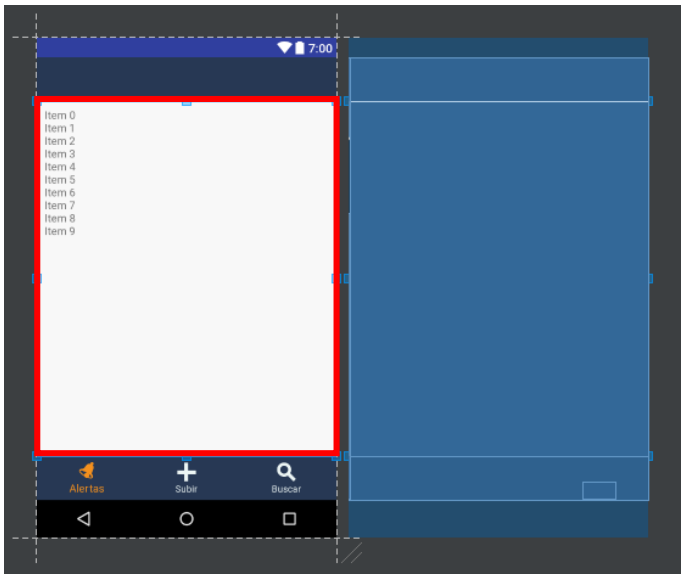


Ilustración 71 - Vista previa del *layout* activity_main.xml

En la ilustración anterior (activity_main.xml) se ve, marcado con un recuadro rojo, el *RecyclerView*. Esta Vista es un *ViewGroup* (Vista que agrupa otras Vistas), es decir; un listado. En la vista previa aparece el listado como si fuera una lista de textos. El Adaptador se encarga de indicarle al *RecyclerView* que debe usar *recipe_card.xml* para cada elemento individual y de esa manera se obtiene una lista con formato de "tarjetas" (estilo Pinterest, Wallapop, etc).

En los archivos de *layouts* se define una Vista de tipo *layout* que actúa de raíz o contenedor y define la disposición de los elementos. Estos *layouts* permiten anidar elementos, es decir; permite meter un *layout* dentro de otro *layout*.

Siguiendo con el mismo ejemplo y obviando tanto el menú inferior como la barra superior, en *activity_main.xml*, se obtiene un *layout* de la siguiente manera:

```
- FrameLayout
    - SwipeRefreshLayout
        - RecyclerView
            - (recipe_card.xml)
```

Este es un ejemplo sencillo que además se ha simplificado. Si se contempla el *layout* de la pantalla de perfil, se observa que se complica un poco más:

```
- LinearLayout
    - FrameLayout
        - ImageView
        - TextView
        - TextView
        - TextView
        - LinearLayout
            - TextView
            - TextView
```

```
            - TextView
        - ImageView
        - View
    - RecyclerView
```

Toda la parte en rojo correspondería a la cabecera. En la práctica, esta parte se ha puesto en un archivo XML separado (*profile_header.xml*) y se referencia desde el *layout*. De tal manera que queda así:

```
- LinearLayout
    - profile_header.xml
    - RecyclerView
```

De esta manera se obtiene un archivo de *layout* más limpio y sencillo de mantener.

En el proyecto se han usado principalmente tres tipos de *layouts*: *LinearLayout, FrameLayout* y *RelativeLayout*.

El primero dispone las Vistas de manera ordenada, una tras otra, dando la opción de hacerlo en vertical o en horizontal. El *FrameLayout* no dispone las Vistas automáticamente sino que de entrada las coloca todas superpuestas y es el desarrollador el que debe recolocarlas. Esto permite la superposición de Vistas. Por último, el *RelativeLayout* permite colocar Vistas respectivamente a otras Vistas (Vista "A" a la derecha de la vista "B" y encima de la vista "C").

12.4. Programación *server-side*

Este proyecto ha requerido la programación de código que se ejecute en el servidor para enviar notificaciones, borrar referencias, actualizar valores de la base de datos y generar miniaturas de las imágenes subidas.

Cuando un usuario sigue a otro usuario, comenta y/o puntúa una receta, se le envía una notificación. Esto es posible gracias a los diferentes *listeners* que se han programado para la base de datos (ver apartado 6.6 Firebase Cloud Functions).

A continuación se explica, a modo de ejemplo, la función llamada *updateScores* encargada de actualizar la puntuación media de una receta y de enviar una notificación. Lo que se encuentra entre llaves (**{}**) se trata como una variable, de tal manera que puede ser cualquier valor.

```
Cuando se produzca un cambio en /recipes/{recipe_id}/score/{user_id}/
    Guarda el valor de /recipes/{recipe_id}/score/
    Entonces:
        Itera sobre todas las parejas clave-valor
        Haz la media de los valores
        Redondea el resultado
        Entonces:
            Guarda el valor de /users/{user_id}/name (nombre)
            Entonces:
                Guarda la media en
                /recipes/{recipe_id}/scoreTotal
                Entonces:
                Guarda la ID del usuario que ha subido la
                receta que se encuentra en
                /recipes/{recipe_id]/user/ (id_usuario)
                Entonces:
                    Guarda el token del usuario que se
                    encuentra en /users/id_usuario/token/
                    Entonces:
                        Construye la notificación
                        Envía la notificación al
                        dispositivo
                        Entonces:
                            Guarda la notificación en la
                            base de datos
```

De manera parecida funcionaría la función llamada *followUser* y *sendNotification*. Ambas mandan una notificación y la guardan en la base de datos, la primera cuando se produce un cambio en /users/{userId}/followers/{followerId}/, es decir; cuando un usuario gana un seguidor. La segunda escucha cambios en /recipes/{recipeId}/comments/{commentId}/ que es cuando una receta recibe un comentario.

Estas funciones que se ejecutan en el servidor están escritas en JavaScript con el nuevo estándar ES6. Gracias a ello se puede usar *JavaScript Promises* [34]y el *callback* [35]*Then()*. Esto es muy interesante ya que gracias a las Promesas se puede ejecutar código de manera asíncrona y especificar un *callback* que se ejecutará únicamente cuando haya acabado (bien o mal) la Promesa. Esto es útil ya que es de interés ejecutar código únicamente cuando se haya ejecutado otro código. En el pseudocódigo anterior se ve claramente en las sentencias “Entonces”.

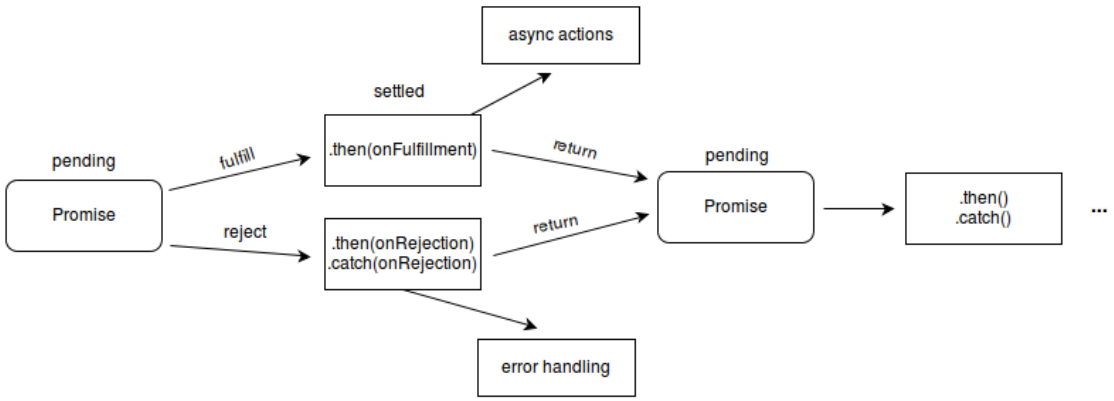


Ilustración 72 - Promesas JavaScript. www.developer.mozilla.org

[34] El objeto *Promise* (Promesa) es usado para computaciones asíncronas. Una promesa representa un valor que puede estar disponible ahora, en el futuro, o nunca. Las promesas en JavaScript representan procesos que ya están sucediendo, y pueden ser encadenados con funciones *callback*. www.developer.mozilla.org

[35] Un *callback* es una “retrollamada”, es decir; en programación se puede hacer una función (A) que reciba como parámetro otra función (B). Cuando se ejecuta A, esta función A ejecutará la función B cuando acabe.

12.5. Evolución

En una primerísima fase del proyecto, antes del desarrollo, se contempló cómo se estructurarían los datos.

Al principio se contempló una estructura de BBDD relacional.

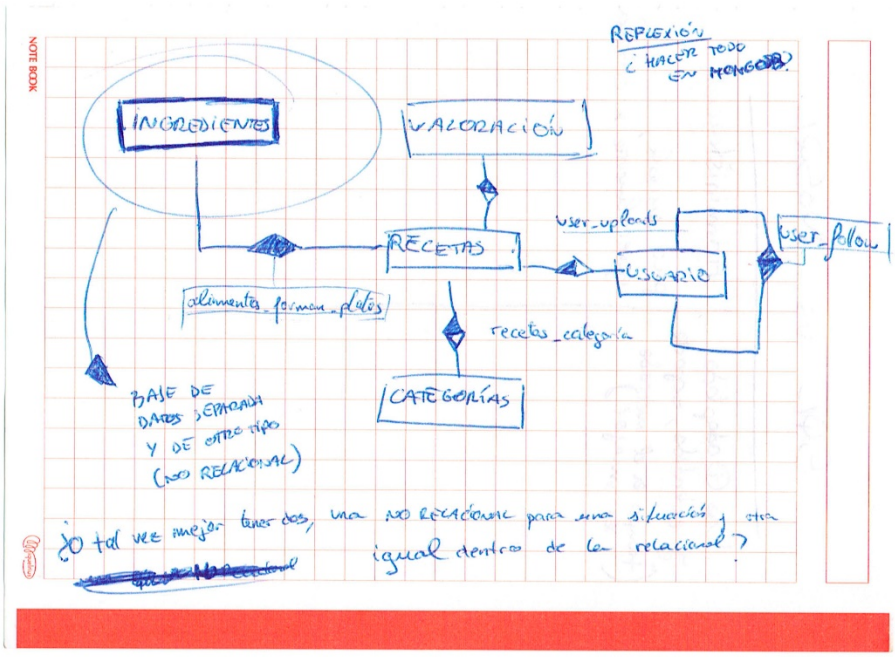


Ilustración 73 - Manuscrito, primer planteamiento de base de datos

No tardó mucho en descartarse completamente este planteamiento de tablas relacionales. Esto fue así porque se buscaba la mayor eficiencia y este tipo de base de datos no ofrecía el rendimiento deseado. Por lo que se empezó a contemplar el uso de una base de datos **no** relacional, es decir; NoSQL.

Para ello se planteó usar **MongoDB**.

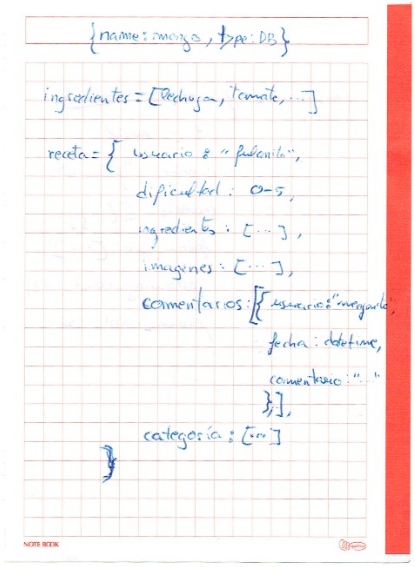


Ilustración 74 - Manuscrito, planteamiento de base de datos no relacional

Más tarde se rechazó la opción de MongoDB por dos simples razones:

- Dificultad de integración con Android
- Existencia de alternativa igual de válida, Firebase

Así pues, se empezó a investigar Firebase y se encontró que ofrecía todo lo que se necesitaba para la app: base de datos no relacional, gestión de usuarios, almacenamiento, notificaciones, etc.

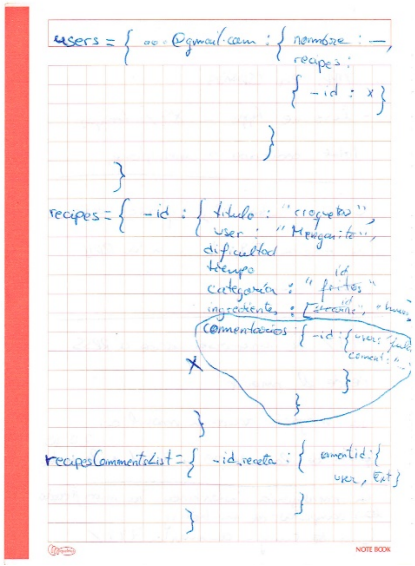


Ilustración 75 - Manuscrito, planteamiento de base de datos en Firebase (I)

Una vez integrado Firebase en el proyecto Android, fueron surgiendo dudas como por ejemplo: ¿cómo obtengo las recetas comunes para diversos ingredientes?

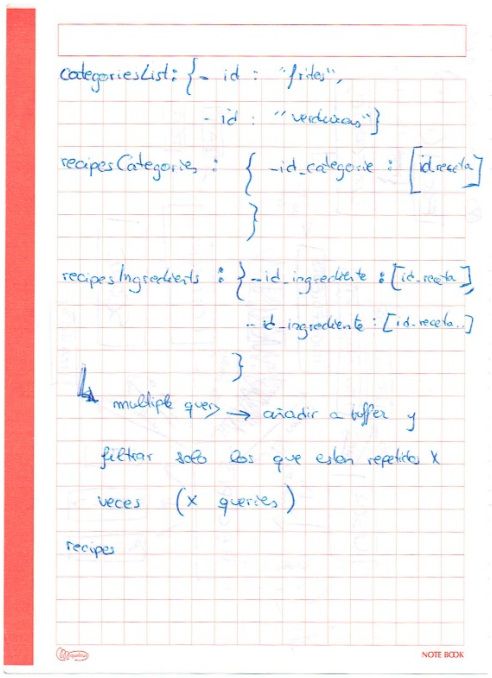


Ilustración 76 - Manuscrito, planteamiento de base de datos en Firebase (II)

Finalmente se definió la estructura de Firebase en formato JSON de la siguiente manera:

```
{
 "users": {
  "user_id": {
   "name": "Alberto",
   "following": {
    "user2_id": true
   },
   "recipes": {
    "recipe1_id": true
   }
  },
  "user2_id": {
   "name": "User 2",
   "following": {},
   "recipes": {}
  }
 },
 "recipes": {
  "recipe1_id": {
   "title": "Primera receta",
   "user": "user_id",
   "difficulty": 2,
   "time": 120,
   "category": {
    "category1_id": true
   },
   "ingredients": {
    "ingredient1_id": true
   },
   "images": {},
   "scores": {
    "user_id": 3,
    "user2_id": 5
   },
   "score_total": 4
  }
 },
 "categories": {
  "category1_id": {
   "name": "Category 1",
   "recipes": {
    "recipe1_id": true
   }
  }
 },
 "ingredients": {
  "ingredient1_id": {
   "name": "Ingredient 1",
   "recipes": {
    "recipe1_id" : true
   }
  }
 }
}
```


Ilustración 77 - Estructura de Firebase

Una vez estructurados los datos, se plantea el flujo de la aplicación para una primera fase.

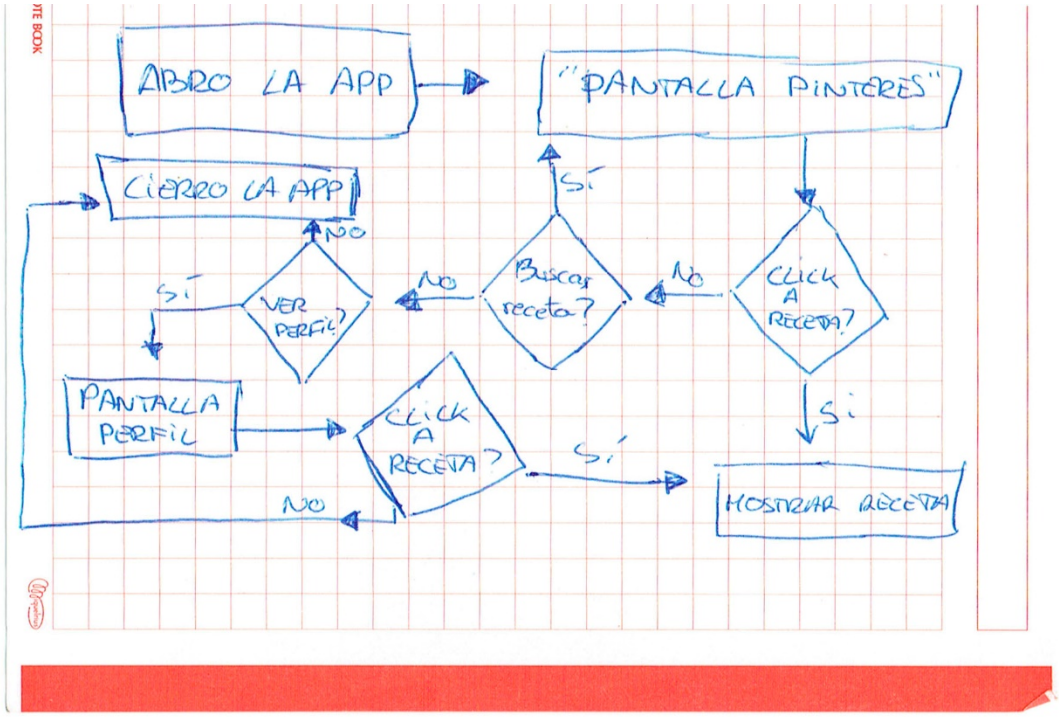


Ilustración 78 - Manuscrito, planteamiento app flow

Lo primero que se propone es mostrar un listado de recetas al abrir la app con un formato “Pinterest”, es decir; cajas con alturas individuales.

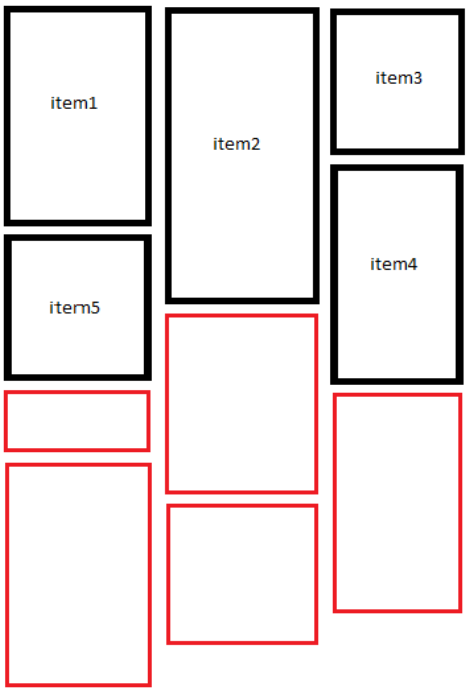


Ilustración 79 - Layout estilo "Pinterest", Staggered Layout

En la pantalla de listado de recetas aparecen en cajitas con una información reducida como el nombre, una imagen y la descripción. En la siguiente imagen se ve el formato “Pinterest” sin el diseño aplicado.

Ilustración 80 - Listado de recetas en formato "Pinterest"

Cuando se toca en la caja de una receta, se navega a la pantalla de detalle:

Ilustración 81 - Pantalla de detalle de receta

Como se observa, se muestran los datos asociados con esa receta.

En la siguiente imagen se observa un formulario sencillo para subir una receta, sin diseño aplicado.

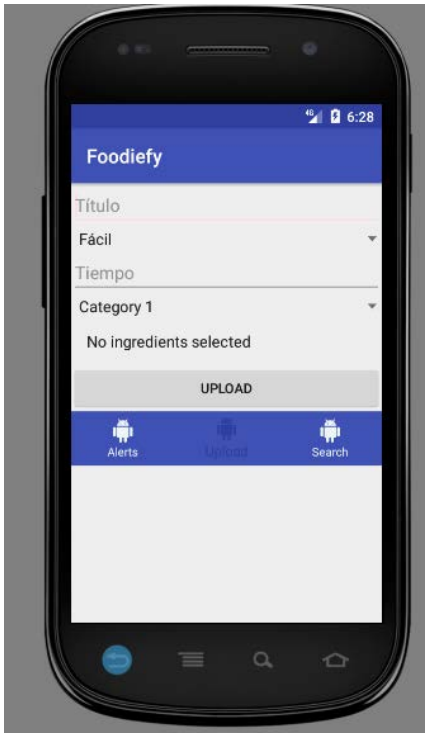


Ilustración 82 - Formulario de subir receta

La siguiente imagen muestra la lógica aplicada en la pantalla de búsqueda por ingredientes; muestra 1 receta encontrada con 1 ingrediente seleccionado (cabrito)

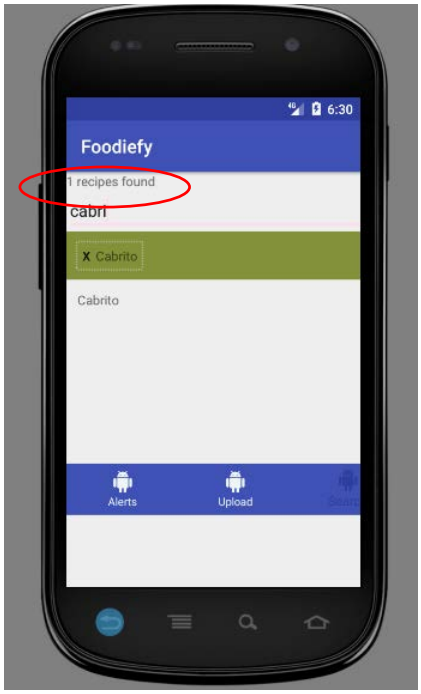


Ilustración 83 - Lógica de la pantalla de búsqueda (I)

Y en la siguiente imagen se observa cómo ahora muestra 0 recetas con los 2 ingredientes seleccionados:

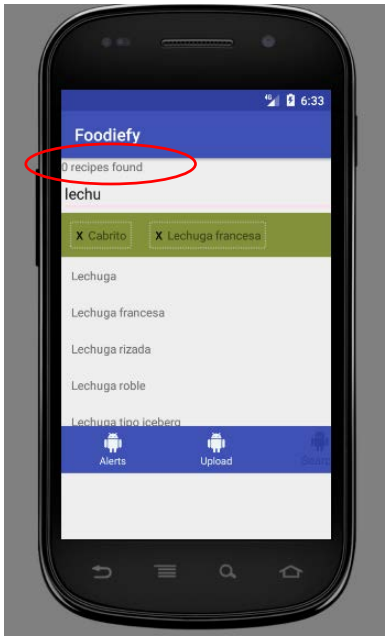


Ilustración 84 - Lógica de la pantalla de búsqueda (II)

A medida que ha ido avanzando el desarrollo se han ido añadiendo características. Por ejemplo, en la siguiente captura se observa un sistema de comentarios y aparece la imagen de usuario. En cambio, la puntuación no existe.

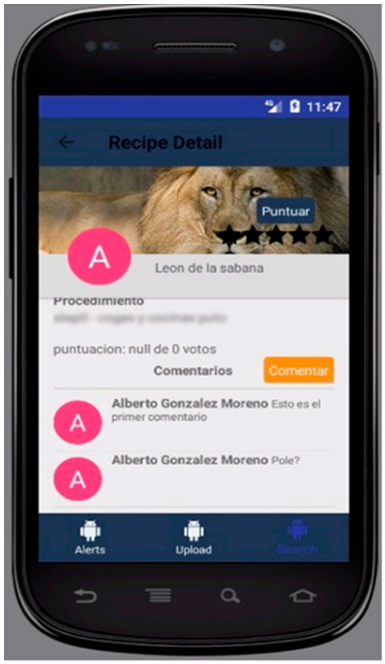


Ilustración 85 - Progreso de la app, adición de comentarios e imagen de usuario

Un poco más adelante en el desarrollo, las recetas ya se empezaban a percibir más bonitas. Ya se muestra la información básica (tiempo, dificultad, comensales), se observa que se puede puntuar e incluso muestra la puntuación actual.

Ilustración 86 - Progreso de la app, información básica de receta y puntuaciones

Una vez conseguida la tarea de recibir notificaciones *push*[36], lo siguiente era poder mostrarlas dentro de la aplicación. Para ello fue menester pensar en una forma mostrarlas debido a que ni Firebase ni Android dispone de una manera de recuperar notificaciones recibidas.

[36] Las Notificaciones Push son mensajes que se envían de forma directa a dispositivos móviles (Smartphones y/o tablets) con sistema operativo iOS, Android, Blackberry y/o Windows Phone. www.wikipedia.org

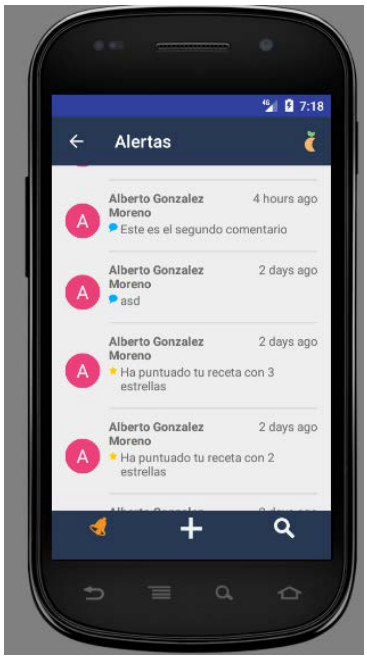


Ilustración 87 - Progreso de la app, notificaciones

Como se ve, no solo se muestran alertas sino también hay una distinción entre el tipo de alerta.

12.6. Landing Page

Como se ha comentado en el apartado 6.7 Firebase Hosting, para la presentación, promoción y difusión de la app, se ha montado una página web estática.

La dirección es la siguiente: www.foodiefyapp.com.

Las tecnologías aplicadas en esta web han sido HTML5, CSS3 y JavaScript. Tiene un diseño *responsive* [37] para su correcta visualización tanto en *mobile* como en *desktop*.

El dominio ha sido contratado a través de los servicios de OVH (www.ovh.es). Para la conexión de este dominio con el servidor web, ha sido necesario añadir tres registros en para el servidor DNS. Estos tres registros son los siguientes:

- Registro de tipo TXT con valor identificativo. Este registro le dice a Firebase que realmente el dominio es de nuestra propiedad, es válido y no está en uso.

Una vez Firebase reconoce el registro anterior en el DNS, se proporciona un certificado SSL para el dominio personalizado y seguidamente se proporciona dos registros

[37] El diseño web responsive o adaptativo es una técnica de diseño web que busca la correcta visualización de una misma página en distintos dispositivos. Desde ordenadores de escritorio a tablets y móviles. www.wikipedia.org

adicionales de tipo A para *conectar* el dominio a la dirección IP correspondiente a los servidores de Firebase.

Finalmente, se añade un subdominio para el redireccionamiento de tal manera que se pueda acceder a la página web poniendo las “www” y sin ponerlas.

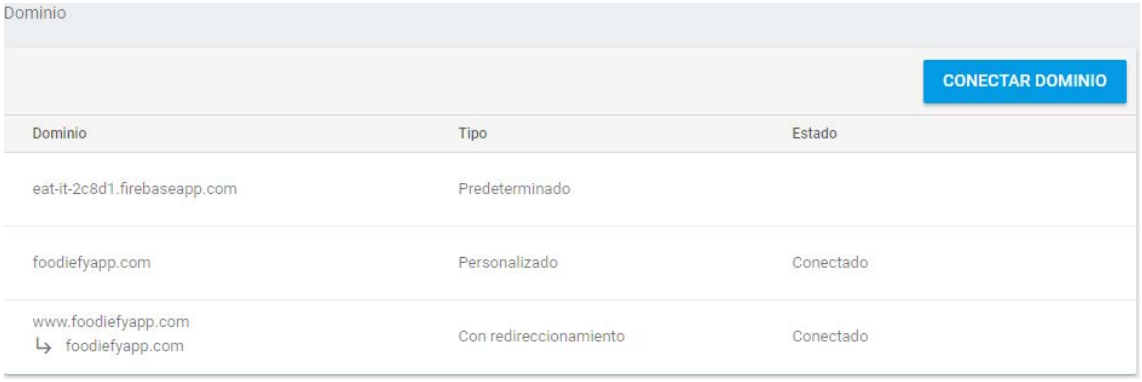


Ilustración 88 - Captura de pantalla de la consola de Firebase mostrando los dominos registrados para el servicio de Firebase Hosting

12.6.1. Posicionamiento

Para el posicionamiento SEO de la página, se han añadido algunos metadatos en el *head* [38] como *keywords* [39]y *description*[40].

También se ha añadido un archivo para especificar *cache* del navegador (y las correspondientes cabeceras HTTP *Cache-Control* y *Last-Modified*[41]*) y* atributos *alt*[42] para las imágenes. Para la correcta indexación por parte de los motores de búsqueda, se ha añadido un archivo *robots* y un mapa del sitio (*sitemap*) *para* habilitar el *crawling* de los buscadores. Para la velocidad de carga se entregan todos los recursos comprimidos y tanto el HTML, el CSS y el JS están *minificados*[43].

En cuanto a la seguridad, la página se carga por defecto mediante el protocolo seguro HTTPS.

[38] Parte de una página web usada por los exploradores para interpretar y mostrar correctamente la página. También es usada para analizarla como hacen los buscadores como Google.
[39] En programación web, las *keywords* son unos valores que van en el [28]*head* y son útiles para posicionamiento web.
[40] Ver [28] *keywords*.
[41] Son dos cabeceras que se envían junto a la página web y que corresponden a la cache del navegador y cómo tratar los recursos de una web como una imagen o una hoja de estilos *css*.
[42] En HTML5 las imágenes pueden llevar un texto alternativo para mostrarlo si la imagen no está disponible o si se usa un lector de pantalla.
[43] La *minificación* de recursos se refiere a la eliminación de bytes innecesarios, como los espacios adicionales, saltos de línea y sangrías. Al minimizar los códigos HTML, CSS y JavaScript, es posible acelerar la descarga, el análisis y el tiempo de ejecución. www.developers.google.com

13. Conclusiones

Sin lugar a dudas este proyecto ha supuesto un constante aprendizaje desde el primer día. No sólo partí de no saber programar Android, sino que además nunca había programado Java. Tal vez por eso cuando se propuso el desarrollo de esta app nos fijamos unas metas muy altas. No obstante, al haber dedicado más horas de las que se esperaba y había planificadas, se ha conseguido acabar satisfactoriamente.

Pese a ello, no sólo se ha logrado alcanzar las metas, sino que además el proyecto me ha hecho interesarme más a fondo en Android y es por ello que ya se está planeando una segunda versión (o primera versión comercial).

Considero que he aprendido mucho y estoy totalmente satisfecho con el resultado obtenido. Ahora que echo la vista atrás puedo concluir que en un inicio no se visualizaba toda la carga de trabajo que suponía este proyecto. Yo mismo al mirar el código que se hizo en fases muy tempranas y compararlo con el de las fases más tardías, puedo ver una progresión y una mejora.

Si tuviera que definir tres fases de desarrollo serían “Make it work, make it good, make it fast” y debido a las limitaciones de tiempo y fechas de entrega, sólo se ha podido abarcar la primera fase: hacer que funcione. Aun así, la aplicación considero que va bastante bien aunque por supuesto es mejorable tanto a nivel de código, UX y fluidez.

Además de lo mencionado, gracias a este proyecto he podido aplicar conocimientos de usabilidad en un escenario real y aprender nuevas técnicas que se usan para mejorar la experiencia de usuario.

14. Agradecimientos

A mis compañeros del trabajo:

Juanlu, *Android Developer*, por ayudarme y resolverme muchísimas dudas.

Yliana, *FrontEnd Developer*, por cedernos el nombre de 'Foodiefy' y darme *feedback* constructivo.

Katia, *Business Analyst*, por ayudarme a analizar el prototipo final y encontrar problemas/mejoras de usabilidad y UX.

Elvis, *Crawler Analyst*, por ayudarme con la auditoría de seguridad.

A Polar por la idea de la app y el apoyo que me ha ofrecido.

A mis profesores Juan José Fábregas y Gabriela Zúñiga por proporcionarme material, realizar el seguimiento y ayudarme con las pruebas de usabilidad.

A toda la gente que no he mencionado, ha probado la app, me ha dado *feedback* y me ha reportado errores durante todo el desarrollo.

15. Índice de figuras

16. Bibliografía

[1] A. Annie, «App Annie 2016 Retrospective», 2016.

[2] Á. Bernal Nicolás, «Proceso de diseño UI/UX», *2014-04-25*. [En línea]. Disponible en: https://es.slideshare.net/AlvaroBernalNicols/proceso-de-diseno-uiux. [Accedido: 15-mar-2017].

[3] C. Buckler, «SQL vs NoSQL: The Differences», *September 18, 2015*, 2015. [En línea]. Disponible en: https://www.sitepoint.com/sql-vs-nosql-differences/. [Accedido: 04-ene-2017].

[4] Ditrendia, «Informe ditrendia: Mobile en España y en el Mundo 2016», 2016.

[5] G. Firebase, «Firebase». [En línea]. Disponible en: https://firebase.google.com/docs/. [Accedido: 11-mar-2017].

[6] Google, «Introduction - Material design - Material design guidelines». [En línea]. Disponible en: https://material.io/guidelines/. [Accedido: 15-mar-2017].

[7] A. Google, «Android - Historia». [En línea]. Disponible en: https://www.android.com/intl/es_es/history/#/marshmallow. [Accedido: 25-abr-2017].

[8] Microsoft, «NoSQL frente a SQL | Microsoft Docs», *November 22, 2016*, 2016. [En línea]. Disponible en: https://docs.microsoft.com/es-es/azure/documentdb/documentdb-nosql-vs-sql. [Accedido: 05-mar-2017].

[9] R. L. Tyrrell, T. G. Townshend, A. J. Adamson, y A. A. Lake, «'I'm not trusted in the kitchen': food environments and food behaviours of young people attending school and college: Table 1», *J. Public Health (Bangkok).*, vol. 38, n.º 2, pp. 289-299, jun. 2016.

[10] A. Vergara, «SQL vs NoSQL ¿Cuál debo usar? - Tech blog for developers | FacilcloudTech blog for developers | Facilcloud», 2016. [En línea]. Disponible en: https://www.facilcloud.com/noticias/sql-vs-nosql-which-one-should-i-use/. [Accedido: 05-mar-2017].

[11] «El boom de las aplicaciones moviles: Retención de users». [En línea]. Disponible en: https://www.yeeply.com/blog/el-boom-de-las-aplicaciones-moviles/. [Accedido: 29-abr-2017].

[12] «Android Nougat: novedades. Android 7.0 Nougat: Multiventana, funciones, análisis - CNET en Español». [En línea]. Disponible en: https://www.cnet.com/es/analisis/google-android-nougat/resena/. [Accedido: 27-abr-2017].

[13] «1983 to today: a history of mobile apps | Media Network | The Guardian». [En línea]. Disponible en: https://www.theguardian.com/media-

network/2015/feb/13/history-mobile-apps-future-interactive-timeline. [Accedido: 26-abr-2017].

[14] «A History of Mobile Apps de AVG Technologies en Prezi». [En línea]. Disponible en: https://prezi.com/rwc6qmvqkrt-/a-history-of-mobile-apps/. [Accedido: 25-abr-2017].

[15] «A Brief History of Mobile App Design - Proto.io Blog». [En línea]. Disponible en: http://blog.proto.io/brief-history-mobile-app-design/. [Accedido: 25-abr-2017].

[16] «java - Most efficient way to return common elements from two string arrays - Stack Overflow». [En línea]. Disponible en: http://stackoverflow.com/questions/8555770/most-efficient-way-to-return-common-elements-from-two-string-arrays. [Accedido: 19-mar-2017].

[17] «Android Distribution Updated for January 2017 - RIP, FROYO! | Droid Life». [En línea]. Disponible en: http://www.droid-life.com/2017/01/10/android-distribution-updated-january-2017-rip-froyo/. [Accedido: 05-mar-2017].

[18] «Android Developers». [En línea]. Disponible en:

https://developer.android.com/. [Accedido: 05-mar-2017].

17. Anexo

17.1. Anexo 1 - Manual de usuario

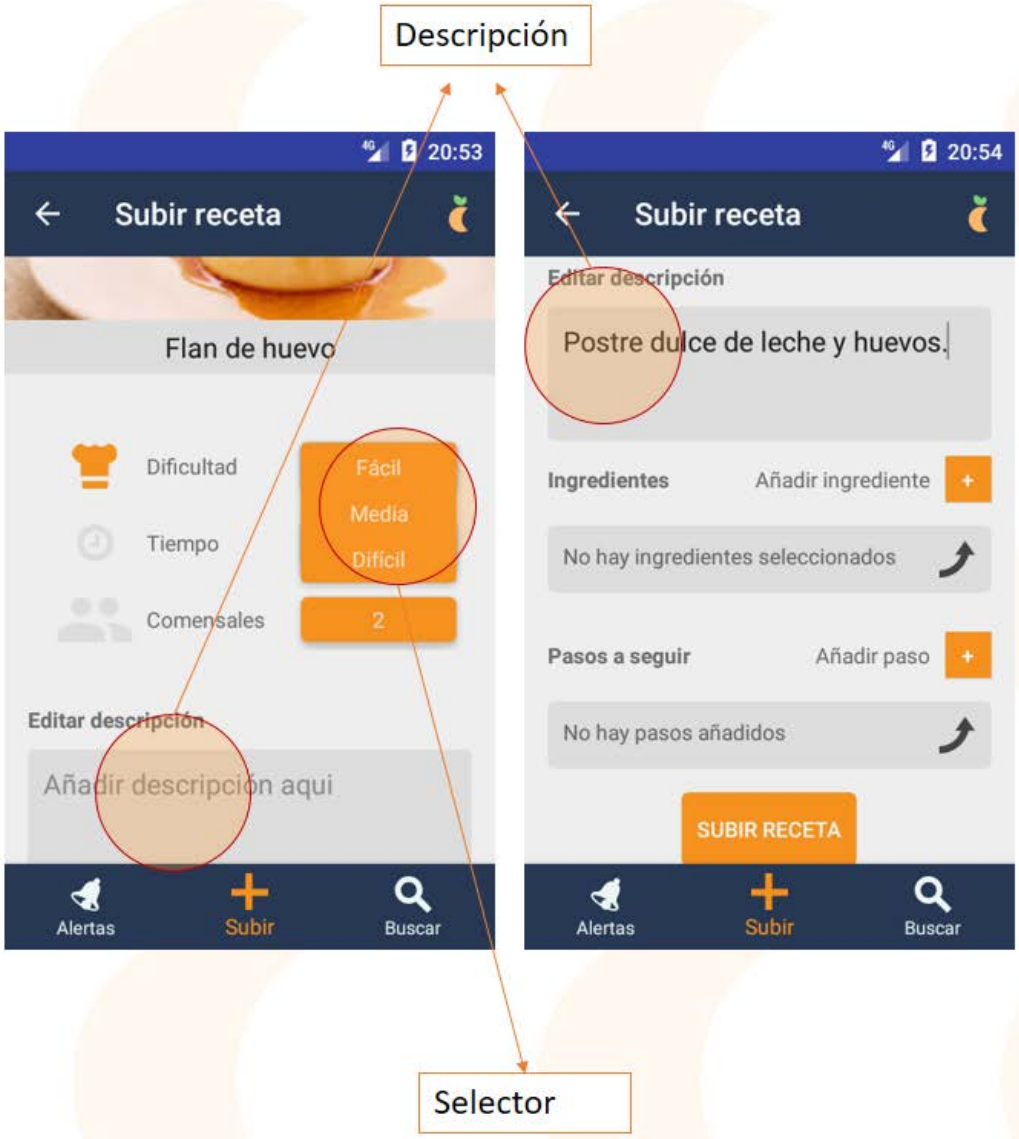
Foodiefy - Manual

44

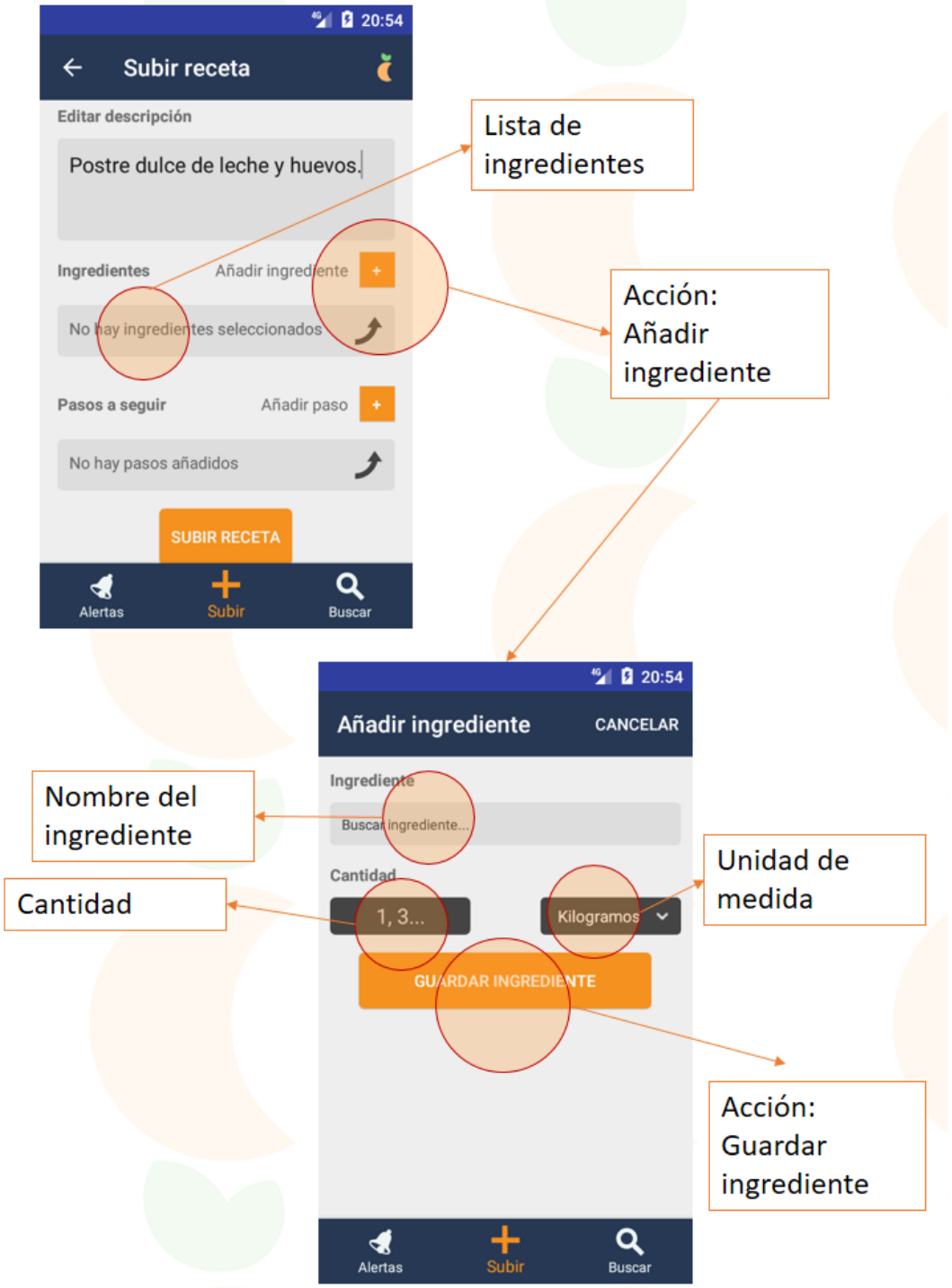
Pantalla principal – Home:
Botón de menú: abre el menú lateral con opciones para el usuario
Receta: receta dentro del listado
Acceso directo al perfil de usuario
Acciones principales presentes durante toda la app

45

Subir receta:
Para subir una receta hay que rellenar el formulario seleccionando una imagen, introduciendo un título, dificultad, tiempo estimado de preparación y número de comensales o raciones.



Subir receta:
La dificultad se selecciona desde un desplegable con tres opciones, tanto el tiempo como el número de comensales es un valor libre (numérico). Es necesario introducir una breve descripción.



47

Subir receta:
Para introducir un ingrediente se debe pulsar sobre el botón de acción: Añadir ingrediente.
Al pulsar sobre este botón se abre una pantalla en la que se debe introducir un ingrediente y seleccionarlo de la lista. También es necesario proporcionar una cantidad y la unidad de medida.
Si el ingrediente introducido no es válido, la aplicación marcará un error y el usuario debe corregir. Puede saltar un error si: el ingrediente no se ha seleccionado de la lista (p.e. si se escribe “Sal” pero no se selecciona de la lista porque no existe), no se ha introducido ningún ingrediente o no se ha introducido una cantidad.

48

Subir receta:
De forma similar, hay que añadir un procedimiento pulsando su correspondiente botón de acción que abre una pantalla para introducir el texto.

49

Subir receta:
Cuando está todo el formulario completo, se puede pulsar el botón de subir receta. Si se completa con éxito se abrirá la pantalla de detalle de receta. Si hay algún error la aplicación marcará la posición donde se encuentra el error (p.e. no se ha introducido el tiempo o no se han añadido ingredientes).

50

Detalle de receta:
En esta pantalla se encuentra toda la información de una receta: imagen, título, características básicas, ingredientes y procedimiento.
Adicionalmente se encuentra un indicador de puntuación y una lista de comentarios con sus correspondientes botones de acción: añadir puntuación y comentario.
Desde la imagen de perfil se puede acceder al perfil del usuario que subió esa receta.

51

Perfil de usuario:
En esta pantalla está la imagen de perfil de usuario, su nombre, las estadísticas (nº de recetas, seguidores y siguiendo) y todas las recetas de ese usuario.
También encontramos un botón de acción: seguir usuario.

52

Alertas:
En esta pantalla hay un listado con las alertas recibidas. Por cada alerta existen varios elementos: nombre e imagen de usuario, tiempo que hace desde que se recibió, un indicador de tipo de alerta y el texto asociado a esa alerta.
Se pueden eliminar arrastrándolas hacia un lado (IRREVERSIBLE).

53

Buscar receta:
Para buscar una receta se deben introducir criterios de búsqueda: ingredientes.
Cuando se escribe el nombre de un ingrediente, se abre debajo una lista para poder seleccionarlo. Cada ingrediente se añade como una etiqueta y se va actualizando el indicador de resultados.

17.2. Anexo 2 - Plan del test de usabilidad (primero)

Foodiefy

Usability Test Plan

Ytarte Roger, González Alberto

April 5th, 2017

Cuando se quieren ver los resultados solo hay que apretar el botón para mostrarlos. Si se quiere eliminar un ingrediente se debe pulsar la “X” de la etiqueta.

17.2.1. Document Overview

This document describes a test plan for conducting a usability test during the development of Foodiefy. The goals of usability testing include establishing a baseline of user performance, establishing and validating user performance measures, and identifying potential design concerns to be addressed in order to improve the efficiency, productivity, and end-user.

The usability test objectives are:

- To determine design inconsistencies and usability problem areas within the user interface and content areas. Potential sources of error may include:
  - Navigation errors – failure to locate functions, excessive keystrokes to complete a function, failure to follow recommended screen flow.
  - Presentation errors – failure to locate and properly act upon desired information in screens, selection errors due to labeling ambiguities.
  - Control usage problems – improper toolbar or entry field usage.
- Exercise the application or web site under controlled test conditions with representative users. Data will be used to access whether usability goals regarding an effective, efficient, and well-received user interface have been achieved.
- Establish baseline user performance and user-satisfaction levels of the user interface for future usability evaluations.

Foodiefy's main target are young people from about 16 to 30 without much cooking skills. Older people are expected to use the app too.

In this first usability test, only the first users group will be involved and about 10 users are expected to participate.

The testing session will occur in a usability lab. The test is expected to be performed on April 5th 2017 in the morning.

17.2.2. Executive Summary

Different points are being evaluated with this test, such as design specific perception items (ex., what do the users understand with an icon?), user's flow clarity (ex., do the users do more keystrokes than expected?) and navigation design errors (ex., do the user know how to go back to the previous screen?). Specific functions to be evaluated are:

- Upload a new recipe

- Search for recipes with the given ingredients
- Check your first recipe comments and give a reply

The user is expected to go to the “upload recipe screen” just by tapping the icon on the bottom navigation menu which is almost always present in the app. The user is also expected to search recipes with the same flow, tap the specific icon on the bottom menu. For the last function, the user is expected to his profile screen first and then look for his first recipe. Once the user has found the recipe, it must be tapped in order to get the recipes details and comments.

Upon review of this usability test plan, including the draft task scenarios and usability goals for Foodiefy, documented acceptance of the plan is expected.

17.2.3. Methodology

For this test, just 10 users are being involved. The users are students from CITM-UPC from any grade. Those students could be app potential users (~16-30 years old without much time/cooking skills).

The test sessions will be set at CITM’s usability lab. The used tool for this test is InVision, a free prototyping software which the users will use to interact with the app’s UI.

The collected information will be:

- Effectiveness during the given tasks (if they have completed the tasks or not).
- Efficiency at the given tasks (how much time the tasks take to the user to be completed).
- The user’s feeling about design (clean interface, good looking items…)
- The user’s mobile skills and cooking skills
- Satisfaction assessment and suggestions.

17.2.3.1. Participants

Ten volunteer students will be recruited from CITM-UPC. They will be around 20-24 years old (the target app is set about 16-30). They are students so they are expected to not have lot of free time in order to learn cooking. They are also expected to be common mobile apps users.

The participants' responsibilities will be to attempt to complete a set of representative task scenarios presented to them in as efficient and timely a manner as possible, and to provide feedback regarding the usability and acceptability of the user interface. The participants will be directed to provide

honest opinions regarding the usability of the application, and to participate in post-session subjective questionnaires and debriefing.

17.2.3.2. Procedure

Participants will take part in the usability test at CITM-UPC's usability lab in Terrassa. A personal computer with the Web site/Web application and supporting software will be used in a typical office environment. The participant's interaction with the Web site/Web application will be monitored by the facilitator seated in the same office. Note will monitor the sessions in the same room.

The facilitator will brief the participants on the Web site/Web application and instruct the participant that they are evaluating the application, rather than the facilitator evaluating the participant. The facilitator will ask the participant if they have any questions.

Participants will complete a pretest demographic and background information questionnaire. The facilitator will explain that the amount of time taken to complete the test task will be measured and that exploratory behavior outside the task flow should not occur until after task completion. At the start of each task, the participant will read aloud the task description from the printed copy and begin the task. Time-on-task measurement begins when the participant starts the task.

The facilitator will instruct the participant to 'think aloud' so that a verbal record exists of their interaction with the Web site/Web application. The note taker will observe and enter user behavior and user comments. He will write down all notes for post-evaluation.

After each task, the participant will complete the post-task questionnaire and elaborate on the task session with the facilitator. After all task scenarios are attempted, the participant will complete the post-test satisfaction questionnaire.

17.2.4. Roles

The roles involved in a usability test are as follows. An individual may play multiple roles and tests may not require all roles.

17.2.4.1. Trainer

- Provide training overview prior to usability testing

17.2.4.2. Facilitator

- Provides overview of study to participants
- Defines usability and purpose of usability testing to participants
- Assists in conduct of participant and observer debriefing sessions
- Responds to participant's requests for assistance

17.2.4.3. Data Logger

- Records participant's actions and comments

17.2.4.4. Test Observers

- Silent observer
- Assists the data logger in identifying problems, concerns, coding bugs, and procedural errors
- Serve as note takers.

Test Participants

- Provides overview of study to participants
- Defines usability and purpose of usability testing to participants
- Assists in conduct of participant and observer debriefing sessions
- Responds to participant's requests for assistance

17.2.4.5. Ethics

All persons involved with the usability test are required to adhere to the following ethical guidelines:

- The performance of any test participant must not be individually attributable. Individual participant's name should not be used in reference outside the testing session.
- A description of the participant's performance should not be reported to his or her manager.

17.2.5. Usability Tasks

The usability tasks were derived from test scenarios developed from use cases and/or with the assistance of a subject-matter expert. Due to the range and extent of functionality provided in the application or Web site, and the short time for which each participant will be available, the tasks are the most common and relatively complex of available functions. The tasks are identical for all participants of a given user role in the study.

The test will occur in a functional prototype, so no database is used. The different presented “screens” will be static and predefined (the same user profile is being used, the same recipes belong to all users…). In the real scenario an Internet connection is required and data is fetched from a real-time database. Although infrastructure is fast, the user could experience little delays caused by connection noise.

The task descriptions below are required to be reviewed by the application owner, business-process owner, development owner, and/or deployment manager to ensure that the content, format, and presentation are representative of real use and substantially evaluate the total application. Their **acceptance is to be documented** prior to usability test.

17.2.6. Usability Metrics

Usability metrics refers to user performance measured against specific performance goals necessary to satisfy usability requirements. Scenario completion success rates, adherence to dialog scripts, error rates, and subjective evaluations will be used. Time-to-completion of scenarios will also be collected.

17.2.6.1. Scenario Completion

Each scenario will require, or request, that the participant obtains or inputs specific data that would be used in course of a typical task. The scenario is completed when the participant indicates the scenario's goal has been obtained (whether successfully or unsuccessfully) or the participant requests and receives sufficient guidance as to warrant scoring the scenario as a critical error.

17.2.6.2. Critical Errors

Critical errors are deviations at completion from the targets of the scenario. Obtaining or otherwise reporting of the wrong data value due to participant workflow is a critical error. Participants may or may not be aware that the task goal is incorrect or incomplete.

Independent completion of the scenario is a universal goal; help obtained from the other usability test roles is cause to score the scenario a critical error. Critical errors can also be assigned when the participant initiates (or attempts to initiate) and action that will result in the goal state becoming unobtainable. In general, critical errors are unresolved errors during the process of completing the task or errors that produce an incorrect outcome.

17.2.6.3. Non-critical Errors

Non-critical errors are errors that are recovered from by the participant or, if not detected, do not result in processing problems or unexpected results. Although non-critical errors can be undetected by the participant, when they are detected they are generally frustrating to the participant.

These errors may be procedural, in which the participant does not complete a scenario in the most optimal means (e.g., excessive steps and keystrokes). These errors may also be errors of confusion (ex., initially selecting the wrong function, using a user-interface control incorrectly such as attempting to edit an un-editable field).

Noncritical errors can always be recovered from during the process of completing the scenario. Exploratory behavior, such as opening the wrong menu while searching for a function, will be coded as a non-critical error.

17.2.6.4. Subjective Evaluations

Subjective evaluations regarding ease of use and satisfaction will be collected via questionnaires, and during debriefing at the conclusion of the session. The questionnaires will utilize free-form responses and rating scales.

17.2.6.5. Scenario Completion Time (time on task)

The time to complete each scenario, not including subjective evaluation durations, will be recorded.

17.2.7. Usability Goals

The next section describes the usability goals for Foodiefy.

17.2.7.1. Completion Rate

Completion rate is the percentage of test participants who successfully complete the task without critical errors. A critical error is defined as an error that results in an incorrect or incomplete outcome. In other words, the completion rate represents the percentage of participants who, when they are finished with the specified task, have an "output" that is correct. Note: If a participant requires assistance in order to achieve a correct output then the task will be scored as a critical error and the overall completion rate for the task will be affected.

A completion rate of 100% is the goal for each task in this usability test.

17.2.7.2. Error-free rate

Error-free rate is the percentage of test participants who complete the task without any errors (critical **or** non-critical errors). A non-critical error is an error that would not have an impact on the final output of the task but would result in the task being completed less efficiently.

An error-free rate of 80% is the goal for each task in this usability test.

17.2.7.3. Time on Task (TOT)

The time to complete a scenario is referred to as "time on task". It is measured from the time the person begins the scenario to the time he/she signals completion.

17.2.7.4. Subjective Measures

Subjective opinions about specific tasks, time to perform each task, features, and functionality will be surveyed. At the end of the test, participants will rate their satisfaction with the overall system. Combined with the interview/debriefing session, these data are used to assess attitudes of the participants.

17.2.8. Problem Severity

To prioritize recommendations, a method of problem severity classification will be used in the analysis of the data collected during evaluation activities. The approach treats problem severity as a combination of two factors - the impact of the problem and the frequency of users experiencing the problem during the evaluation.

17.2.8.1. Impact

Impact is the ranking of the consequences of the problem by defining the level of impact that the problem has on successful task completion. There are three levels of impact:

- High - prevents the user from completing the task (critical error)
- Moderate - causes user difficulty but the task can be completed (non-critical error)
- Low - minor problems that do not significantly affect the task completion (non-critical error)

17.2.8.2. Frequency

Frequency is the percentage of participants who experience the problem when working on a task.

- High: 30% or more of the participants experience the problem
- Moderate: 11% - 29% of participants experience the problem
- Low: 10% or fewer of the participants experience the problem

17.2.8.3. Problem Severity Classification

The identified severity for each problem implies a general reward for resolving it, and a general risk for not addressing it, in the current release.

Severity 1 - High impact problems that often prevent a user from correctly completing a task. They occur in varying frequency and are characteristic of calls to the Help Desk. Reward for resolution is typically exhibited in fewer Help Desk calls and reduced redevelopment costs.

Severity 2 - Moderate to high frequency problems with moderate to low impact are typical of erroneous actions that the participant

recognizes needs to be undone. Reward for resolution is typically exhibited in reduced time on task and decreased training costs.

Severity 3 - Either moderate problems with low frequency or low problems with moderate frequency; these are minor annoyance problems faced by a number of participants. Reward for resolution is typically exhibited in reduced time on task and increased data integrity.

Severity 4 - Low impact problems faced by few participants; there is low risk to not resolving these problems. Reward for resolution is typically exhibited in increased user satisfaction.

17.2.9. Reporting Results

The Usability Test Report will be provided at the conclusion of the usability test. It will consist of a report and/or a presentation of the results; evaluate the usability metrics against the pre-approved goals, subjective evaluations, and specific usability problems and recommendations for resolution. The recommendations will be categorically sized by development to aid in implementation strategy. The report is anticipated to be delivered to the Project UCD Contact by April 17th 2017.

17.3. Anexo 3 - Resultados del test de usabilidad (primero)

Foodiefy Test

Alberto González

April 17th, 2017

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

17.3.1. Introduction

Foodiefy is a social network app that serves as a cooking recipe encyclopedia. Foodiefy lets the users interact with other users through the recipes they upload. It also lets performing an accurate recipe search based on specific criteria.

A usability test is intended to determine the extent an interface facilitates a user's ability to complete routine tasks. Typically the test is conducted with a group of potential users either in a usability lab, remotely (using e-meeting software and telephone connection), or on-site with portable equipment. Users are asked to complete a series of routine tasks. Sessions are monitorized and analyzed to identify potential areas for improvement to the web site.

The Foodiefy developers conducted an onsite usability test using high-fidelity prototype located on the test administrator's laptop. The two developers acted as test administrators, data loggers, introductory… all roles. The developers were present at the session and took notes about the results they were obtaining.

17.3.2. Executive Summary

The Foodiefy project team conducted an onsite usability test at the HCI laboratory at CITM-UPC on April 5th, 2017. The purpose of the test was to assess the usability of the app interface design, icons identification and get any usability issues.

Ten students participated in the test. Each individual session lasted approximately ten-twelve minutes. All participants had to perform the same routine tasks.

In general all participants found the Foodiefy's interface to be clear, straightforward and 100% thought the application was easy to use (80% scored the interface's beauty and ease as 4-5 in a scale of 5). 100% of the participants were familiar with mobile apps and social networks.

The test identified only a few minor problems which are:

- The upload recipe button ("+" icon) was not always identified immediately.
- On the search screen participants thought they could write whatever they wanted ("carbonara", "tortilla de patatas", "postres").
- The notifications icon is confused as a profile or friends icon.

This document contains the participant feedback, task completion rates, ease or difficulty of completion ratings, errors, and recommendations for improvements. A copy of the scenarios and questionnaires are included in the Attachments' section.

17.3.3. Methodology

17.3.3.1. Sessions

The developers contacted and recruited participants in situ from the university.

Each individual session lasted approximately ten minutes. During the session, the test administrator explained the test session and asked the participant to fill out a brief background questionnaire (see Attachment A). Then, the tasks were explained to the participants, consecutively.

After the three tasks, the administrator asked the participant to rate the interface, the overall impression on a 5-point Likert Scale with measures ranging from Strongly Disagree to Strongly Agree. Participants were also asked that whom they would recommend the app or whatever they wanted to say about it. Post-task scenario subjective measures included (see Attachment B):

- How easy it was to use the app.
- Intuitiveness about the navigation.
- Understanding the menu items.
- Look&feel

17.3.3.2. Participants

All participants were students at the CITM-UPC.

Ten of the ten participants (100%) completed the test. The participants were 50% male 50% female. The 90% were 16-21 years old, 5% were 22-25 years old and the last 5% were 26-32 years old.

17.3.3.3. Evaluation Tasks/Scenarios

Test participants attempted completion of the following tasks:

- Upload a recipe
- Find and answer a comment a user made on the uploaded recipe.

- Perform a recipe search.

17.3.4. Results

17.3.4.1. Task Completion Success Rate

All participants successfully completed Task 1 (upload a recipe), Task 2 (answer a comment) and Task 3 (perform a search).

Task Completion Rates

Participant		Task 1	Task 2	Task 3
1		√	√	√
2		√	√	√
3		√	√	√
4		√	√	√
5		√	√	√
6		√	√	√
7		√	√	√
8		√	√	√
9		√	√	√
10		√	√	√
Success		10	10	10
Completion Rates		100%	100%	100%

17.3.4.2. Errors

Foodiefy's developers captured the number of errors participants made while trying to complete the task scenarios.

A non-critical error is an error that does not prevent successful completion of the scenario.

All participants completed all tasks. 10% made a non-critical error in Task 1, and 20% in Task 2. Task 3 was completed without any error.

17.3.4.3. Summary of Data

The table below displays a summary of the test data. Low completion rates and satisfaction ratings and high errors and time on tasks are highlighted in red.

Summary of Completion, Errors

Task	Task Completion	Errors
1	10	1
2	10	2
3	10	0

17.3.4.4. Overall Metrics

17.3.4.4.1. Overall Ratings

After task session completion, participants rated the site for eight overall measures (See Attachment B). These measures include:

- How easy it was to use the app.
- Intuitiveness about the navigation.
- Understanding the menu items.
- Look&feel
- Interface beauty

Most of the participants (90%) agreed (i.e., agree or strongly agree) that the interface is easy and intuitive. All the participants (100%) agreed that the app is beautiful and easy to use. The 90% of the participants understood the menu items and had a good overall impression. The majority of participants (77%) agreed they would use Foodiefy.

See table below.

Post-Task Overall Questionnaire

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Percent Agree
App interface is beautiful				5	5	100%
App interface is intuitive			1	4	5	90%

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Percent Agree
Navigation is easy and intuitive			1	5	4	90%
Apps is easy to use				5	5	100%
Understanding of the menu items			1	4	5	90%
Overall impression (Very Bad – Very Good)			1	4	5	90%

**Percent Agree (%) = Agree & Strongly Agree Responses combined*

17.3.4.4.2. Likes, Dislikes, Participant Recommendations

Upon completion of the tasks, participants provided feedback for what they liked most and least about the website, and recommendations for improving the website.

Liked Most

The following comments capture what the participants liked most:

- App main idea
- Ease of using

Liked Least

The following comments capture what the participants liked the least:

- App main color

Recommendations for Improvement

- More ways to search recipes.
- Ability to change measurement units.
- Private messages.
- Categories.
- Search results filtering.

17.3.5. Recommendations

The recommendations section provides recommended changes and justifications driven by the participant success rate, behaviors, and comments. Each recommendation includes a severity rating. The following recommendations will improve the overall ease of use and address the areas where participants experienced problems or found the interface/information architecture unclear.

Upload a recipe (Task 1)

Task 1 required the participants to tap on the upload recipe icon and fill the information.

Change	Justification	Severity
• Enlarge and/or change the color of the "+" icon. • Make clearer the procedure input when uploading a recipe and differentiate it from the description.	The 10% of the participants made a non-critical error during this task because they didn't recognize the icon at first. Some participants filled the recipe description as it was the procedure step by step.	High

Reply a comment (Task 2)

Task 2 required participants to find the notification's icon and inside this section find the specific notificacion.

Change	Justification	Severity
• Change the notifications icon.	About 70% of the participants recognized the icon only because the small circle indicating how many notifications they had. Without the number they thought it was a profile or friends section instead of notifications.	High

Search a recipe (Task 3)

Task 3 required the user to search a recipe using the search engine.

Change	Justification	Severity
• Add a previous screen to chose the search method.	In order to be clear what the user is required to search, the previous screen lets the user chose what to search: recipes by ingredients, recipes by 'free text input', other users, etc.	High

17.3.6. Conclusion

Implementing the recommendations and continuing to work with users (i.e., real lay persons) will ensure a continued user-centered website.

Most of the participants found Foodiefy to be well-organized, comprehensive, clean and uncluttered, very useful, and easy to use. Having a centralized app to find recipes is key to many if not all of the participants. Implementing the recommendations and continuing to work with users (i.e., real lay persons) will ensure a continued user-centered app.

17.3.7. Attachments

17.3.7.1. Attachment A – Background questionnaire

Edad:

- 15 o menos
- 16-21
- 22-26
- 26-32
- 33 o más

Género

- Hombre
- Mujer

¿Qué experiencia tienes con aplicaciones móviles?

- Excelente, uso apps a diario.
- Bien, uso apps a menudo. (por ejemplo: Descargas una app para un uso puntual)
- Mal, uso apps muy de vez en cuando. (por ejemplo: WhatsApp y Facebook)
- Fatal, nunca uso apps. (por ejemplo: Sólo llamo y recibo llamadas)

¿Qué experiencia tienes con las redes sociales?

- Pertenezco a más de una red social y entro a diario
- Sólo estoy en una red social y entro de vez en cuando
- Sólo estoy en una red social y casi nunca entro
- ¡Jamás me encontrarás en Internet!

¿Qué experiencia tienes en la cocina?

- ¡Soy todo un chef!
- Bastante bien (Hago unos espaguetis ricos, ricos… y con fundamento)
- Me defiendo (El arroz con pollo me queda de lujo)
- Fatal (Se me rompen los huevos fritos…)

¿Compartes trucos, consejos, recetas, etc. con tus conocidos?

- Sí
- No

¿Cuánto tiempo dedicas a preparar la comida?

- Todo el necesario para que quede bien rico.
- Lo justo para que no esté crudo.
- Casi nada o nada, prácticamente no cocino.

En función a la respuesta anterior, responde por qué:

17.3.7.2. Attachment B – User satisfaction questionnaire

¿Crees que has completado todas las tareas satisfactoriamente?

- Sí
- No

La interfaz de la aplicación me ha parecido bonita y estética

Muy en desacuerdo Muy de acuerdo

1	2	3	4	5

La interfaz de la aplicación me ha parecido intuitiva

Muy en desacuerdo Muy de acuerdo

1	2	3	4	5

La navegación me ha parecido fácil e intuitiva

Muy en desacuerdo Muy de acuerdo

1	2	3	4	5

En relación al prototipo de Foodiefy, la he encontrado fácil de usar.

Muy en desacuerdo Muy de acuerdo

1	2	3	4	5

He entendido rápidamente la función de los elementos del menú.

Muy en desacuerdo Muy de acuerdo

1	2	3	4	5

Mi impresión general respecto a Foodiefy es:

Muy mala Muy buena

1	2	3	4	5

¿Usarías Foodiefy?

- Sí
- No

En caso afirmativo, ¿Para qué? (Se admiten múltiples respuestas)

- Subir recetas
- Buscar recetas
- Encontrar a los mejores chefs

¿A quién le recomendarías esta app?

Edad:

Suieto:

Aquí puedes decir lo que quieras respecto a Foodiefy. Si te ha encantado puedes dejar constancia aquí, si te ha defraudado también. Aceptamos cualquier crítica, sugerencia o comentario ☺